Emilia Pulliainen

Intraoraaliröntgensensorien testauslaitteen LabVIEW-ohjelmointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

17.5.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Emilia Pulliainen Intraoraaliröntgensensorien testauslaitteen LabVIEW-ohjelmointi 49 sivua 17.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Ohjelmistosuunnittelija Thomas von Schulmann Lehtori Antti Liljaniemi

Tämän insinöörityön tavoitteena oli kehittää hammashoitoteknologiayritys Planmeca Oy:n valmistamien ProSensor-intraoraalisensorien testauslaitteen ohjelmistoa. Intraoraalisensorit ovat hampaiden röntgenkuvauksessa käytettäviä potilaan suuhun asetettavia digitaalisia ilmaisimia. Sensorien testauksen tarkoituksena on varmistaa, että sensori toimii oikein ja täyttää laatuvaatimukset.

Testauslaitteen ohjelmistokehitysprojektissa hyödynnettiin Scrum-projektinhallintamenetelmää. Projektille varattu aika jaettiin lyhyempiin jaksoihin, sprintteihin, joiden aikana ryhmälle kunkin sprintin alussa jaetut tehtävät pyrittiin saamaan valmiiksi. Ohjelmistokehitykseen käytettiin LabVIEW-ohjelmointikieltä. Kuvaprosessointia vaativissa tehtävissä hyödynnettiin lisäksi NI Vision Development Modulea, joka sisältää LabVIEW-noodeja sekä konenäköohjelma Vision Assistantin. Osa testauslaitteen ohjelmista oli aikaisemmin tehty MATLAB-kielellä ja käännettiin LabVIEW'lle.

Insinöörityössä käsitellään neljää testauslaitteelle projektin aikana tehtyä LabVIEW-ohjelmaa, jotka liittyvät testauslaitteen ja tietokoneen väliseen kommunikointiin sekä intraoraalisensorien testikuvien analyysiin. Työn alussa tehtiin ohjelma RS-232-sarjaporttien väliseen kommunikointiin sekä kaksi ohjelmaa, jotka kommunikoivat testauslaitteen kanssa psComm-apuohjelman välityksellä. Näiden ohjelmien avulla voidaan ohjata sensorin tilaa, sekä lukea ja kirjoittaa sensorin parametreja, kuten sensorin sarjanumero.

Intraoraalisensorien testauksessa sensorilla otetaan testikuvia, jotka käyvät läpi joukon analyysejä. Insinöörityön aikana toteutettiin LabVIEW'lle yksi näistä analyyseistä, Klusterianalyysi, joka analysoi kuvan sisältämiä virheellisiä pikseleitä ja pikseliryhmittymiä, niiden määrää ja kokoa sekä määrittää, täyttääkö sensori laatuvaatimukset. Ohjelman toteutuksessa hyödynnettiin Vision Development Modulea, jota ilman ohjelman toteuttaminen olisi ollut huomattavasti vaikeampaa.

Työn lopputulos on kolme onnistuneesti toteutettua pienempää ohjelmaa ja yksi laajempi kuva-analyysiohjelma, jotka tulevat olemaan osa testauslaitteen ohjelmistoa. Testauslaitteen ohjelmiston kehitys jatkui vielä insinöörityöprojektin päättymisen jälkeen.

Avainsanat	LabVIEW, NI Vision, röntgenkuvantaminen

Author(s) Title Number of Pages Date	Emilia Pulliainen Intraoral X-ray Sensor Testing Device Programming Using LabVIEW 49 pages 17 May 2017
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialisation option	
Instructor(s)	Thomas von Schulmann, Software Developer Antti Liljaniemi, Senior Lecturer

The aim of this thesis was to develop software for a device used for testing intraoral X-ray sensors manufactured by Planmeca Oy. Intraoral sensors are digital detectors used in dental imaging for capturing X-ray images from inside the patient's mouth. The goal of testing is to confirm that the sensor works as intended and meets quality specifications.

The Scrum agile software development framework was utilized during the project. The time allotted for the project was divided into shorter periods called sprints, during which each member of the programming team aimed to complete the tasks agreed at the beginning of each sprint. The LabVIEW programming language was used for software development. In addition, the NI Vision Development Module, consisting of LabVIEW nodes and the machine vision software Vision Assistant, was used for tasks requiring image processing. Some pre-existing testing device programs made using MATLAB were converted to LabVIEW during the project.

Four LabVIEW programs developed for the testing device are discussed in this thesis. Programs that control RS-232 serial communication between the computer and the testing device and allow the user to control the status of the sensor and read/write parameters, such as serial number, were created at the beginning of the project.

Test images taken during the sensor testing procedure are subjected to several analyses, of which one (Cluster Analysis) was implemented as a LabVIEW program as part of this thesis project. Cluster Analysis searches for invalid pixels and pixel clusters, compares their number and size to limits and determines based on the results whether the sensor requires quality control or should be rejected. This program made use of the Vision Development Module, without which implementing some parts of the program in LabVIEW would have been difficult.

The outcome of the project is three smaller programs and one larger image analysis program that will be part of the testing device software. The software development project continues after the completion of the thesis work.

Keywords	LabVIEW, NI Vision, X-ray imaging

Sisällys

Lyhenteet ja määritelmät

CAD/CAM	Computer-Aided Design / Computer-Aided Manufacturing. Tietokoneavusteinen suunnittelu / valmistus.
CCD	Charge-Coupled Device. Digitaalisessa kuvantamisessa käytettävä valoherkän kennon tyyppi.
CDC	USB Communications Device Class. USB-laiteluokka kommunikaatiolaitteille.
CMOS	Complementary Metal Oxide Semiconductor. Mikropiiriteknologia, jota hyödynnetään muun muassa valoherkissä kennoissa.
FPGA	Field Programmable Gate Array. Uudelleen ohjelmoitava digitaalinen mikropiiri.
KKTT	Kartiokeilatietokonetomografia. Kolmiulotteinen röntgenkuvantamismenetelmä.
NI	National Instruments. Amerikkalainen ohjelmistoja ja laitteita tuottava yritys.
PSP	Photostimulable Phosphor Plate. Röntgenkuvantamisessa käytettävä kuvailmaisintyyppi.
subVI	Nimitys LabVIEW-aliohjelmalle.
TFT	Thin-Film Transistor. Transistorityyppi, jossa puolijohdemateriaali on ohuena kalvona kiinteällä pinnalla.
VI	Virtual Instrument, virtuaali-instrumentti. Nimitys LabVIEW-ohjelmalle.
VISA	Virtual Instrument Software Architecture. Ohjelmointirajapinta, jota hyödynnetään laitteen ja tietokoneen välisessä kommunikoinnissa.

1 Johdanto

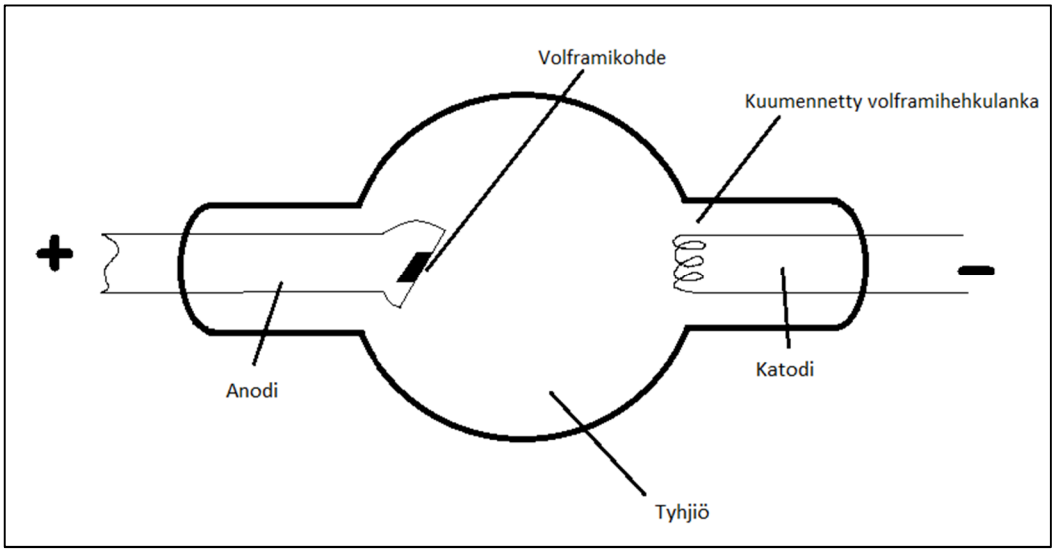
Tämän opinnäytetyön tavoitteena on kehittää suomalaisen hammashoitoteknologiayritys Planmeca Oy:n valmistamien ProSensor-intraoraalisensorien testauslaitteen ohjelmistoa. Intraoraalisensorit ovat hampaiden röntgenkuvauksessa potilaan suuhun asetettavia CCD- tai CMOS-teknologiaan perustuvia ilmaisimia, jotka muuntavat sensoriin osuvan röntgensäteilyn ensin valoksi ja edelleen sähköiseksi signaaliksi, joka muutetaan digitaaliseen muotoon. Planmeca Oy:ssä intraoraalisensorien testauslaitteen ja sen ohjelmiston avulla varmistetaan, että sensori toimii oikein ja täyttää sille asetetut laatuvaatimukset.

Osa testauslaitteen ohjelmistosta on jo aiemmin toteutettu MATLAB-kielellä. Planmeca Oy on kuitenkin siirtymässä käyttämään yhä enemmän LabVIEW'tä, joka on National Instrumentsin kehittämä graafinen ohjelmointikieli. Insinöörityöprojektin tavoitteena on testauslaitteen MATLAB-ohjelmien kääntäminen tietyiltä osin LabVIEW-muotoon sekä uusien LabVIEW-ohjelmien kehitys testauslaitteelle. Insinöörityön puitteissa toteutettavat LabVIEW-ohjelmat tulevat olemaan osa laajempaa ohjelmistoa, jolla ohjataan testauslaitetta, analysoidaan testikuvia sekä lopulta määritetään, täyttääkö sensori laatuvaatimukset. LabVIEW'n lisäksi projektissa hyödynnetään NI Vision -konenäköohjelmistoa. Työskentely tapahtuu osana työryhmää ja ohjelmistokehityksen tehostamiseksi projektissa hyödynnetään Scrum-projektinhallintamenetelmää, joka kuuluu ns. ketteriin ohjelmistokehityksen menetelmiin.

Opinnäytetyössä käsitellään aluksi työn taustaa (luku 2), kuten röntgenkuvantamisen teoriaa ja teknologiaa sekä Planmeca Oy:tä yrityksenä. Seuraavaksi (luku 3) esitellään työssä käytetyt ohjelmointi- ja ohjelmistotyökalut. Kappaleessa 4 käsitellään opinnäytetyön varsinaisia tuloksia eli testauslaitteeelle kehitettyjä LabVIEW-ohjelmia. Lopuksi tarkastellaan työn tuloksia sekä projektin aikana opittua (luku 5).

2 Työn tausta

2.1 Röntgensäteily ja sen tuottaminen



Kuva 1. Röntgenputken rakenne. Mukaillen [3].

Röntgensäteily on korkeaenergistä sähkömagneettista säteilyä aallonpituusalueella 0,01–10 nanometriä [1, s. 3]. Röntgensäteilyn löysi vuonna 1895 saksalainen Wilhelm Röntgen, joka palkittiin löydöstään Nobelin fysiikan palkinnolla (1901) [2].

Röntgensäteilyä voidaan tuottaa röntgenputkeksi kutsutun lasisen tyhjiöputken avulla, joka sijaitsee röntgenlaitteessa putkipään sisällä [4, s. 16]. Röntgenputken rakenne on esitetty kuvassa 1. Röntgenputkessa elektronilähteenä toimii kuumennetusta volframihehkulangasta koostuva katodi. Anodi koostuu kuparikappaleeseen kiinnitetystä volframikohteesta. Katodia kuumennetaan ja sen ympärille muodostuu elektronipilvi. Putken päiden välinen suuri potentiaaliero saa elektronit liikkumaan nopeasti anodin volframikohdetta päin. Keskittävä laite kohdistaa elektronit volframikohteeseen. Elektronien törmätessä volframiin vapautuu lämpöä (~99 %) ja röntgensäteilyä (~1 %). Anodin kupariosa ja röntgenputkea ympäröivä öljy absorboivat suurimman osan vapautuvasta lämmöstä. Röntgensäteilyä syntyy, kun elektronit hidastuvat kulkiessaan volframiatomiytimien läheisyydestä ja menettävät energiaa; vapautuvaa röntgensäteilyä kutsutaan jarrutussäteilyksi. Tämän lisäksi röntgensäteilyä syntyy kiihdytettyjen elektronien törmätessä volframin sisäkuoren elektroneihin saaden ne virittymään (eksitaatio) tai irtoamaan

atomeista (ionisaatio); tätä säteilyä kutsutaan ominaissäteilyksi. Putkipäätä ympäröi lyijykotelo, jossa olevasta aukosta kulkeva röntgensäteily muodostaa röntgensäteen, jota käytetään röntgenkuvantamisessa. [4, s. 16–20.]

2.2 Röntgenkuvantaminen

Röntgensäteily on korkeaenergistä säteilyä ja kykenee läpäisemään ihmisen kudokset, minkä ansiosta sitä voidaan käyttää lääketieteellisessä kuvantamisessa. Röntgenkuvantamisessa kuvattava kohde asetetaan röntgenlähteen ja kohteen läpäisevän säteilyn tallentavan kuvailmaisimen väliin. Hammaslääketieteellisessä kuvantamisessa ilmaisin voi olla potilaan suussa, jolloin puhutaan intraoraalikuvantamisesta, tai suun ulkopuolella, mitä kutsutaan ekstraoraalikuvantamiseksi. [4, s. 12.] Perinteisesti röntgenkuvan tallentamiseen on käytetty radiografista filmiä. Erilaisten digitaalisten ilmaisimien käyttö on kuitenkin yleistynyt ja ne ovat korvaamassa filmiä kuvailmaisimena [4, s. 3].

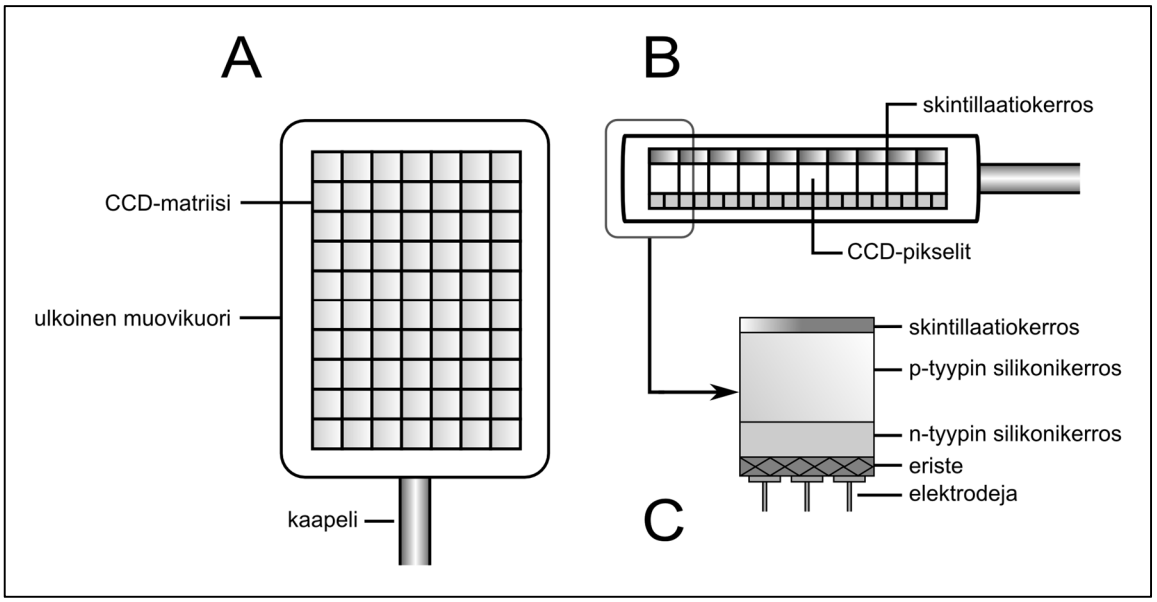
Filmipohjaisessa röntgenkuvantamisessa kuva syntyy kun röntgensäteet läpäisevät potilaan ja kulkeutuvat filmille muodostaen latentin kuvan. Kun röntgensäteily tunkeutuu kudokseen, osa siitä absorboituu tai hajaantuu; siihen, kuinka suuri osa säteilystä läpäisee kohteen, vaikuttavat muun muassa kohteen tiheys ja paksuus sekä röntgensäteen intensiteetti [4, s. 5, 22]. Mitä enemmän säteilyä pääsee kudoksen läpi filmille, sitä tummempi alue kuvassa on. Filmille tallentunut latentti kuva saadaan näkyviin kemiallisen prosessin kautta. [1, s. 4.]

Radiografinen filmi voidaan jakaa kahteen tyyppiin, suoraan ja epäsuoraan toimivaan filmiin. Suoraan toimiva radiografinen filmi reagoi enimmäkseen röntgenfotoneihin. Tätä filmityyppiä käytetään, kun tarvitaan korkeaa kuvanlaatua ja anatomisten yksityiskohtien erottaminen on tärkeää. Epäsuoraan toimiva radiografinen filmi reagoi röntgenfotoneiden sijaan pääasiassa valofotoneihin, ja sitä käytetään yhdessä kahden röntgenvahvistuslevyn (engl. *intensifying screen*) kanssa. Röntgenvahvistuslevyt koostuvat *fluorescent phospor* -materiaalista, joka reagoi röntgenfotoneihin vapauttamalla valofotoneita. Koska yksi röntgenfotoni synnyttää monta valofotonia, epäsuorasti toimivan filmin herkkyys on parempi kuin suoraan toimivan filmin, minkä ansiosta potilaan altistusta röntgensäteilylle voidaan vähentää. Syntyvät valofotonit kuitenkin leviävät filmillä suhteellisen suurelle

alueelle, minkä vuoksi epäsuorasti toimivan filmin resoluutio on huonompi kuin suoraan toimivan filmin. [4, s. 31–35.]

Digitaalisen röntgenkuvantamisen perusperiaate on sama kuin filmipohjaisessa röntgenkuvantamisessa: kuva määräytyy kohteen läpi kulkeneen röntgensäteilyn määrän mukaan [4, s. 3]. Digitaalinen ilmaisin muuttaa röntgensäteilyn analogiseksi signaaliksi. Muuntotapa vaihtelee ilmaisintyypin mukaan; jotkin ilmaisimet muuttavat säteilyn suoraan sähköiseksi signaaliksi, toiset taas muuttavat säteilyn ensin valoksi. Analoginen signaali muunnetaan digitaaliseksi A/D-muuntimen avulla ja siirretään tietokoneeseen prosessointia varten. [1, s. 7.]

Hammaslääketieteessä käytettäviä digitaalisia ilmaisimia ovat CCD- tai CMOS-pohjaiset sensorit, tasopaneeli-ilmaisimet (engl. *flat panel detector*) sekä PSP-kuvalevyt (engl. *photostimulable phospor storage plate)* [1, s. 103–108; 4, s. 38–43]. CCD/CMOS-sensoreihin tai tasopaneeleihin pohjautuvaa kuvantamista kutsutaan suoraksi radiografiaksi (engl. *direct radiograpy,* DR), kun taas kuvalevyjä käytettäessä puhutaan tietokoneradiografiasta (engl. *computed radiography*, CR) [5, s. 4].



Kuva 2. CCD-pohjaisen intraoraalisensorin perusrakenne. A. Sensorin kuvantamispinta edestä kuvattuna. B. Sensori sivulta kuvattuna. CCD-pikselien päällä on skintillaatiokerros, joka muuntaa röntgensäteilyn valoksi. C. Yksittäisen pikselin rakenne. Pikselit koostuvat p- ja n-tyypin silikonista. Mukaillen [4, s. 40].

CCD-/CMOS-pohjaiset intraoraalisensorit koostuvat piipohjaisesta pikselimatriisista sekä tarvittavasta elektroniikasta muovisessa kotelossa. Useimmissa on johto tiedonsiirtoa varten, joskin myös langattomia sensoreita on olemassa. Tähän luokkaan kuuluvat ekstraoraalisensorit ovat tyypillisesti CCD-pohjaisia ja sisältävät pitkän, vain muutaman pikselin levyisen pikselimatriisin, jolla kuvattava kohde skannataan yhdessä sensorin kanssa linjatun kapean röntgensäteen kanssa. [4, s. 39–41.]

CCD-sensori (engl. *charge-coupled device*) on P- ja N-tyypin piistä koostuvien pikselien muodostama matriisi (kuva 2). Pikseleiden päällä on skintillaatiokerros, joka muuttaa siihen osuvan röntgensäteilyn valoksi. Valon osuminen piihin synnyttää kuhunkin pikseliin varauksen, jotka yhdessä muodostavat varauskuvion. Varauskuvio luetaan siirtämällä rivin varaus riviltä seuraavalle ja lopulta vahvistimeen, josta analoginen signaali etenee kaapelia pitkin A/D-muuntimeen. CMOS-sensori (engl. *complementary metal oxide semiconductor*) muistuttaa rakenteeltaan CCD-sensoria, mutta pikseleiden varaukset luetaan eri tavalla. Pikselit ovat erillään toisistaan ja kukin pikseli on yhdistetty transistoriin, josta varaukset luetaan yksitellen. [1, s. 40.] Planmecan ProSensor-intraoraalisensorit (ks. osio 2.4) ovat CMOS-pohjaisia [6].

Tasopaneeli-ilmaisin (engl. *flat-panel detector*) koostuu yksittäisiä pikseleitä vastaavista detektorielementeistä, jotka sisältävät ohutkalvotransistorin (engl. *thin-film transistor*, TFT), kondensaattorin sekä havaitsevan osan, joka detektoi röntgen- tai valofotoneja riippuen tasopaneelin tyypistä [1, s. 110]. Tasopaneelit jaetaan epäsuoraan ja suorasti toimiviin tasopaneeleihin riippuen niiden rakenteesta ja röntgensäteilyn käsittelytavasta [1, s. 106].

Epäsuoraan toimivassa tasopaneelissa röntgensäde osuu cesiumjodidista (CsI) tai gradoliniumoksisulfidista (Gd_2O_2S) koostuvaan skintilaattoriin, joka muuttaa röntgensäteilyn valoksi. CsI-kristallit muodostavat neulamaisia, röntgensäteen suuntaisesti kulkevia rakenteita. Tämän tyyppistä skintilaattoria kutsutaan rakenteelliseksi. Röntgensäteilyn kulkiessa rakenteellisen skintillaattorin läpi tapahtuu vain vähän sivusuuntaista hajontaa. Gd_2O_2S-kristallit ovat partikkeleina ilman säännöllistä rakennetta, muodostaen rakenteettoman skintillaattorin jossa tapahtuu paljon sivusuuntaista hajontaa. Skintillaattorin jälkeen valofotonit kulkevat amorfisesta piistä (a-Si) koostuvaan fotodiodikerokseen, joka muuttaa valon sähkövarauksiksi. Fotodiodikerroksen vieressä on varausten lukua varten ohutkalvotransistori sekä kondensaattoreita, jotka varastoivat fotodiodissa

syntyneet sähkövaraukset. [1, s. 108–109.] Suoraan toimivassa tasopaneelissa röntgensäteily osuu amorfisesta seleenistä (a-Se) koostuvaan valoa johtavaan kerrokseen (engl. *photoconductor*), jonka alla on TFT-matriisi. Päällimmäisen elektrodin ja TFT-komponenttien väille syntyy sähkökenttä. Röntgensäteiden osuminen seleenikerokseen synnyttää sähkövarauksia jotka liikkuvat kohti TFT-elementtejä, mm. kondensaattoreihin jotka tallettavat varaukset. [1, s. 109–110, 113.]

PSP-kuvalevyihin (engl. *Photostimulable phospohor plate*) kuuluu intraoraali- ja ekstraoraalipaneeleita. PSP-kuvalevyt on päällystetty materiaalilla joka absorboi ja varastoi siihen osuneen säteilyn energiaa. Säteilytetty kuvalevy asetetaan lukulaitteeseen, joka skannaa kuvalevyn laserilla vapauttaen sen varastoiman energian valona. Käyttökertojen välillä kuvalevy pyyhitään intensiivisellä valolla. [1, s. 42–43.]

Kaksiulotteisten röntgenkuvien tulkinta voi joissakin tapauksissa olla ongelmallista ja niiden tarjoama informaatio rajallista tai jopa harhaanjohtavaa. Tiheämpi kohde voi esimerkiksi peittää taakseen muita kohteita, ja kohteen muodon ja sijainnin hahmottaminen voi olla hankalaa. [1, s. 7.] Uudemmat menetelmät mahdollistavat nykyään kuitenkin myös kolmiulotteisten röntgenkuvantamisen. Yksi 3D-kuvantamismenetelmä on kartiokeilatietokonetomografia (KKTT). KKTT-laitteen säteilylähde tuottaa kartion tai pyramidin muotoisen röntgensäteilykeilan. Säteilylähde on samassa kuvaustelineessä ilmaisimen kanssa, joka on yleensä tasopaneeli. Kuvauksen aikana laite pyörähtää 180–360 astetta kohteen ympäri ja ottaa suuren määrän kaksiulotteisia röntgenkuvia eri kulmista, kun taas tavallisessa tietokonetomografiassa (TT) otetaan leikkeitä. Kuville suoritetaan matemaattinen esikäsittely, jonka jälkeen niistä kootaan 3D-tilavuusdataa, jota voidaan tarkastella halutusta suunnasta. [7.]

2.3 Planmeca Oy

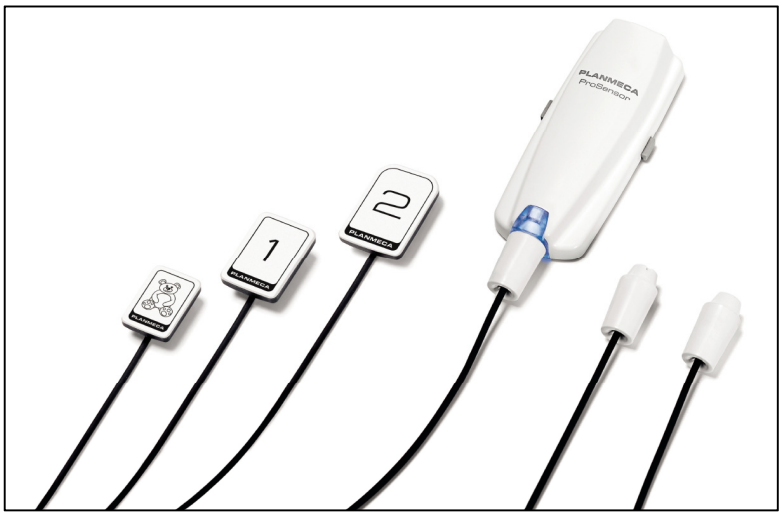
Planmeca Oy on Heikki Kyöstilän vuonna 1971 perustama yritys, joka suunnittelee ja valmistaa 2D- ja 3D-röntgenlaitteita, hammashoitokoneita sekä kuvantamis- ja CAD/CAM -ohjelmistoja [8; 9]. Planmeca Oy on kuudesta yhtiöstä koostuvan Planmeca Group -yhtiöryhmän emoyhtiö.

Planmeca Groupiin kuuluvat seuraavat yhtiöt [9]:

- Planmeca Oy
- Planmed Oy: Suunnittelee ja valmistaa kuvantamislaitteita ja tarvikkeita mammografiaan ja ortopediseen kuvantamiseen.
- Plandent Oy: Hammashoitoalan tarvikkeiden, laitteiden ja ohjelmistojen toimittaja ja tukipalveluiden tarjoaja. Osa suurempaa Plandent-liiketoimintaryhmää.
- LM-Instruments Oy: Suunnittelee ja valmistaa muun muassa hammaslääketieteen käsi-instrumentteja, ultraäänilaitteita ja oikomishoidon tuotteita.
- Opus Systemer AS: Kehittää hammashoitopraktiikan ohjelmistoja.
- Triangle Furniture Systems Inc.: Suunnittelee kaapistoja ja sterilisointikeskuksia hammashoitoon.

2.4 ProSensor-intraoraalisensorien testaus Planmeca Oy:ssä

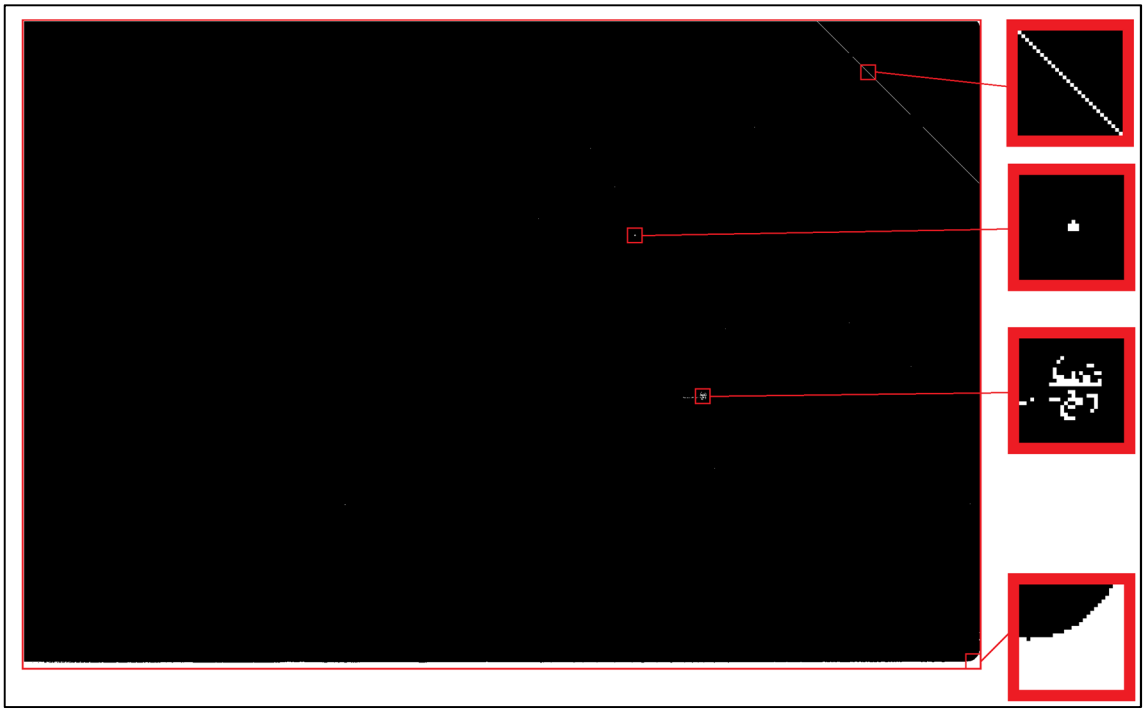
ProSensor HD on Planmecan kehittämä intraoraalisensori, joka sisältää CMOS-kennon ja skintillaattorin, joka konvertoi siihen osuvan röntgensäteilyn valoksi [10; s. 11]. Sensoria saa kolmessa koossa (0, 1 ja 2; kuva 3). Sensorin ohjausrasiassa on LED-valo, joka antaa informaatiota kuvantamisen etenemisestä. ProSensor HD voidaan liittää Planmeca ProX -intraoraaliröntgenlaitteeseen ja se on saatavilla Ethernet- tai USB-liitännällä [11].



Kuva 3. Planmeca ProSensor HD -intraoraalisensoreita kolmessa eri koossa (0, 1 ja 2) sekä sensoreiden ohjausyksikkö. Lähde: Planmecan materiaalipankki [12].

Ennen intraoraalisensorin toimittamista asiakkaalle on varmistettava, että laite toimii oikein ja täyttää laatuvaatimukset. Planmeca Oy:ssä vaurioiden aiheuttamat ja muut toimintahäiriöt haetaan tähän tarkoitukseen rakennetulla testauslaitteella. Laitteen runko koostuu lyijyllä vuoratusta puisesta laatikosta. Laatikon päässä on röntgensäteilyä tuottava putkipää ja laatikon päällä putkipäätä kontrolloiva laitteisto. Laatikon toisessa päässä olevaan kehikkoon asetetaan testattava intraoraalisensori, joka on liitetty ohjausyksikköön.

Testattavalla sensorilla otetaan testikuvia (kuva 4), jotka käyvät läpi joukon analyysejä, joissa saaduista kuvista muun muassa etsitään viallisia pikseleitä. Reunoja siirretään sisäänpäin, jolloin suurin osa huonoista pikseleistä poistuu ja loput haetaan tähän tarkoitukseen kehitetyllä ohjelmalla. Huonot pikselit voivat pahimmillaan peittää kuvasta tärkeitä yksityiskohtia.



Kuva 4. Esimerkki testikuvasta. Oikealla suurennettuna esimerkkejä kuvassa olevista virheellisistä pikseleistä sekä kuvan reuna-alueesta. Kuvaa on muokattu erilaisten virheiden havainnollistamiseksi samassa kuvassa.

2.5 Scrum

Testauslaitteen ohjelmistokehitysprojektissa hyödynnettiin työskentelyn tehostamiseksi Scrum-projektinhallintamenetelmää. Scrum on ketterän ohjelmistokehityksen menetelmä, jolla helpotetaan monimutkaisten projektien hallintaa [13]. Projektin johtaja oli käynyt Scrum master -koulutuksen. Projekti jaettiin sprintteihin eli ajanjaksoihin, jonka aikana projektin työntekijät pyrkivät saamaan sprintin alussa sovitut tehtävät valmiiksi. Sprintin aikana pidettiin päivittäin lyhyitä aamukokouksia, joissa käytiin läpi päivän tavoitteet, tehtävien edistyminen, esitetiin kysymyksiä tehtävistä, sekä sovittiin mahdollisista tapaamisista. Kunkin sprintin lopuksi pidettiin kokous, jossa Jira-järjestelmään merkittiin työtehtävien valmistuminen tai pidennettiin tehtävän vaatimaa aikaa.

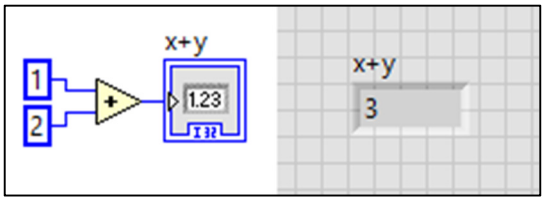
3 Työssä käytetyt ohjelmistot

3.1 LabVIEW

LabVIEW on National Instrumentsin luoma kehitysympäristö, jolla voidaan luoda muun muassa datan keräämiseen, käsittelyyn, analysoitiin sekä laitteiden ja instrumenttien ohjaamiseen käytettäviä ohjelmia [14, s. 1]. LabVIEW pohjautuu G-kieleen, joka on graafinen ohjelmointikieli. G-kielen graafinen luonne helpottaa ohjelman rakenteen hahmottamista. LabVIEW-ohjelmassa data kulkee johtoja pitkin noodeiksi kutsuttujen objektien välillä. Noodit voivat olla funktioita, aliohjelmia tai rakenteita, joihin kuuluvat esimerkiksi for-silmukka, while-silmukka ja case-rakenne. [15.] Noodi suorittaa operaation heti kun se on saanut tarvittavan datan. Tätä toimintatapaa kutsutaan datavirtaohjelmoinniksi (engl. *dataflow programming*). [14, s. 38.]

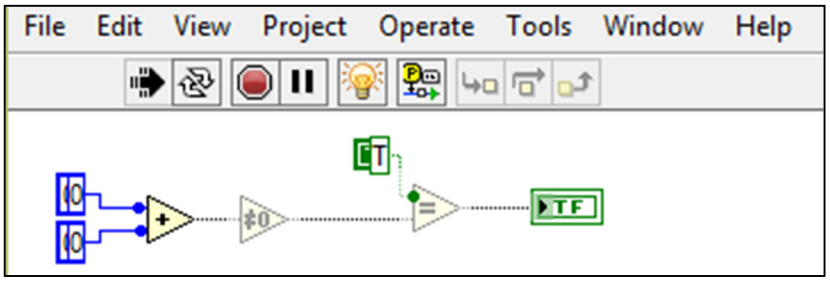
LabVIEW-ohjelmia kutsutaan virtuaali-instrumenteiksi (engl. *virtual instrument*, VI). Kullakin VI:llä on etupaneeli (engl. *front panel*), joka edustaa graafista käyttöliittymää. Etupaneelilla sijaitsevat kontrollit, esimerkiksi painikkeet, liukusäätimet ja muut elementit joiden avulla ohjelmaa hallitaan, sekä indikaattorit, esimerkiksi kuvaajat, LED:it ja tekstikentät, jotka esittävät ohjelman tuottamaa tai keräämää dataa. Lohkokaavio (engl. *block diagram*) puolestaan sisältää VI:n ohjelmointielementit, kuten noodit, johdot, terminaalit ja subVI:t. [15.] SubVI:t ovat LabVIEW'n vastine tekstipohjaisten ohjelmointikielten aliohjelmille. SubVI:lla on tavallisen VI:n tapaan etupaneeli ja lohkokaavio, mutta ne toimivat toisen VI:n sisällä. Valitusta koodista on mahdollista luoda subVI "Create SubVI" -käskyllä koodin selkeyttämiseksi. Lisäksi mitä tahansa tallennettua VI:tä on mahdollista käyttää subVI:nä lisäämällä se haluttuun kohtaan toisessa VI:ssa. [16.] Etupaneelin kontrollit ja indikaattorit näkyvät lohkokaaviossa terminaaleina [15].

Kuvassa 5 vasemmalla on esimerkki yksinkertaisesta LabVIEW-koodista. Kolmio, jonka sisällä on +-merkki on noodi, jonka funktio on laskea yhteen siihen syötettävät kaksi numeerista vakiota (1 ja 2). Laskutoimituksen tulos näytetään etupaneelilla indikaattorissa (kuva 5, oikea); lohkokaaviossa indikaattori näkyy terminaalina.



Kuva 5. Yksinkertainen yhteenlaskuohjelma LabVIEW'ssä.

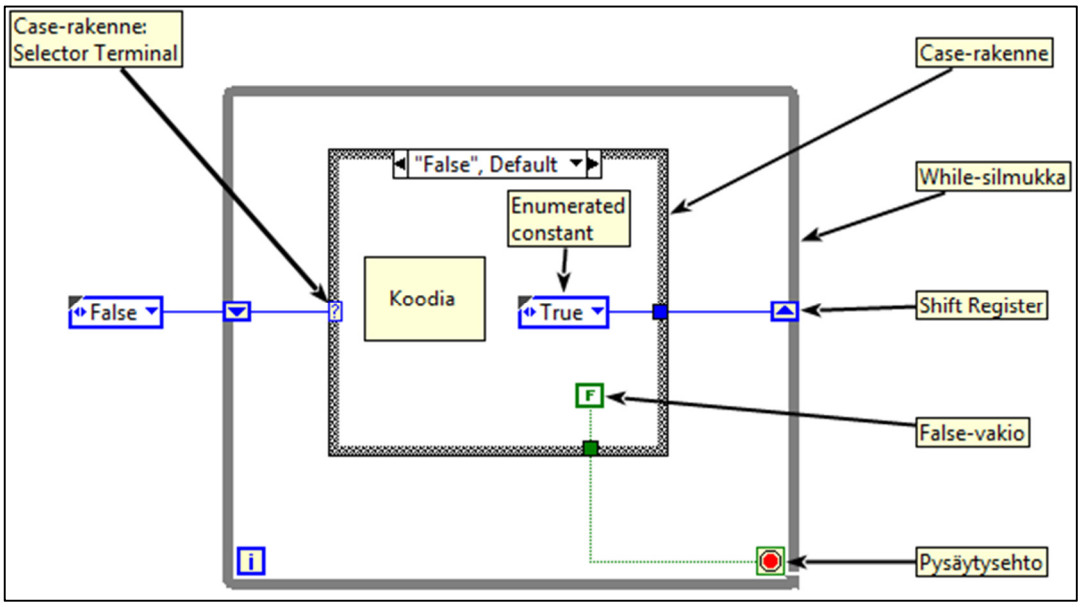
LabVIEW'ssä datan kulkua vedettyjä johtoja pitkin voidaan visualisoida Highlight Execution -toiminnolla, joka korostaa kulloinkin aktiivisena olevan ohjelmakoodin [17]. Kuvassa 6 Highlight Execution on päällä. Harmaat johdot ja haaleat noodit eivät ole vielä saanet tarvitsemaansa dataa. Johdot joilla on tyyppiään vastaava väri (kuvassa sininen, numeerinen) ovat jo kuljettaneet dataa. Värikkäitä johtoja pitkin kulkevat pallot kuvaavat datan kulkua ohjelmassa.



Kuva 6. Highlight Execution -toiminto.

3.1.1 State machine -arkkitehtuuri

State machine on LabVIEW-ohjelmointiarkkitehtuuri, jota ohjelmoija voi käyttää ohjelman perustana. State machine -pohjaisessa ohjelmassa on tiloja (engl. *state*), joiden välillä ohjelma siirtyy käyttäjän käskyjen tai ohjelman sisältämien ohjeiden perusteella [18]. State machine -rakennetta hyödynnettiin laajalti tässä työssä toteutetuissa LabVIEW-ohjelmissa.



Kuva 7. State machine -rakenne.

State machinen pohjarakenne koostuu seuraavista elementeistä (kuva 7):

- While-silmukka, joka pyörittää sisältämäänsä ohjelmakoodia kunnes saa pysäytysehtoa vastaavan arvon [18].

- Shift register pitää kirjaa seuraavaksi aktivoitavasta tilasta. Lohkokaavioissa Shift Register esitetään kahtena silmukan oikeassa ja vasemmassa sivussa sijaitsevana laatikkona, joiden sisällä on nuoli (kuva 7). Shift registereitä käytetään datan siirtoon silmukan iteraatiosta toiseen. Initialisoimaton shift register säilyttää siihen tallennetun arvon kunnes VI suljetaan. Shift register initialisoidaan syöttämällä siihen dataa, jonka tyypin shift registerit omaksuvat. Silmukan käynnistyessä ensimmäisen kerran vasen shift register tulostaa siihen tallennetun datan. Silmukan sisällä on seuraava syöttö, joka tallennetaan oikean puoleiseen shift registeriin. Seuraavan iteraation alussa oikean puoleisen shift registeriin tallennettu data siirtyy vasemman puoleiseen shift registeriin, joka tulostaa sen silmukan käynnistyessä. Siirrettävän datan tulee olla saman tyyppistä. [19.]

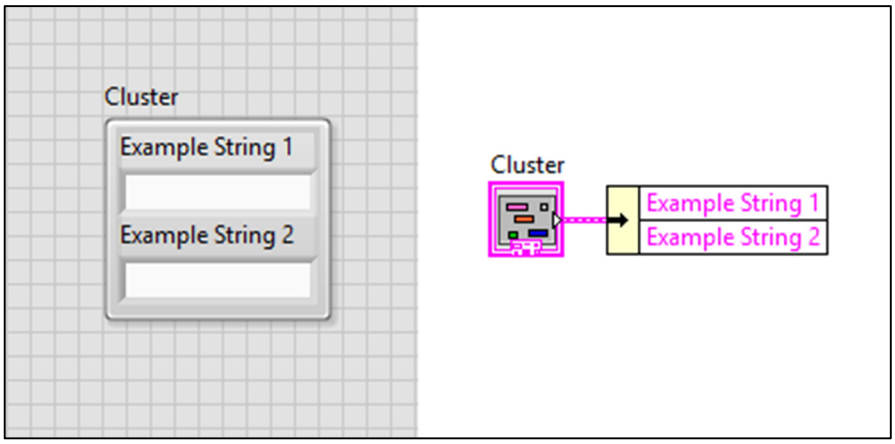
- Case-rakenne sisältää yhden tai useamman tilan, joista yksi ajetaan kun case-rakenne saa tilaa vastaavan syötteen. Case-rakenteen vasemmassa sivussa on case-terminaali (engl. *case terminal*; kuvassa 7 laatikon sisällä oleva kysymysmerkki), johon syötetään ajettava tila. [20.]

- Case-valitsija (engl. *case selector*) määrittää seuraavan aktiivisen tilan. Case-valitsijan muoto voi vaihdella; kuvassa 7 case-valitsijana toimii Enumerated constant, jolla voi olla esimerkissä arvo true tai false [21].

Kuvan 7 esimerkissä vasemmanpuoleinen shift register initialisoidaan false-arvolla [19]. While-silmukan ensimmäisessä iteraatiossa vasen shift register syöttää tilan nimen

case-terminaalin. Case-rakenne aktivoi case-terminaalin sisältämää arvoa vastaavan tilan. Tilan sisältämä ohjelmakoodi suoritetaan. Seuraavaksi aktiivisen tilan sisältämä case-valitsija tulostaa seuraavan tilan nimen, joka kulkee johtoa pitkin oikean puoleiseen shift registeriin. Silmukan kertauksen alussa vasemmanpuoleinen shift register palauttaa viimeisimmän tallennetun arvon. Arvo kulkee johtoa pitkin case-terminaaliin, joka kertoo uuden tilan case-rakenteelle. Ohjelma jatkaa tilasta toiseen kunnes while-silmukka saa lopetuskäskyn. [21.]

3.1.2 Klusteri



Kuva 8. Klusteri etupaneelissa ja Unbundle by Name -noodi lohkokaaviossa.

Klusteri on LabVIEW'ssä datarakenne, jolla voidaan ryhmittää eri tyyppistä dataa [14, s. 104]. Klustereita voidaan luoda ja muokata Bundle-funktioilla. Klusterin sisältämään dataan päästään käsiksi Unbundle-funktioilla. Tässä työssä klusterirakennetta hyödynnettiin Klusterinanalyysissa (osio 4.4) useiden eri tietojen tallentamiseen ja siirtämiseen tilasta toiseen.

3.1.3 NI-VISA

National Instruments VISA (Virtual Instrument Software Architecture) on ohjelmointirajapinta (*engl. Application Programming Interface*, API), joka kehitettiin parantamaan yhteentoimivuutta eri valmistajien mittauslaitteiden välillä. NI-VISA:n käyttö vähentää kehitys- ja testaussysteemien rakentamiseen kuluvaa aikaa. NI-VISA:a on mahdollista käyttää kommunikointiin eri instrumentointiväylien kanssa, muun muassa GPIB, US, Serial ja Ethernet. VISA:lla luotua ohjelmaa voidaan käyttää muilla LabVIEW'tä tukevilla

sovellusalustoilla. [22.] Tässä opinnäytetyössä NI-VISA:n virtuaali-instrumentteja käytettiin RS-232-kommunikointiohjelmassa (osio 4.3).

3.2 NI Vision

NI Vision pitää sisällään kolme National Instrumentsin ohjelmistopakettia konenäkösovelluksiin: Vision Builder for Automated Inspection, Vision Development Module ja Vision Acquisition Software. Vision Builder for Automated Inspection on itsenäinen konenäön kehitysympäristö, joka ei vaadi käyttäjältä ohjelmointitaitoja. Vision Development Module puolestaan sisältää kuvankäsittely- ja konenäköfunktioita LabVIEW-, C/C++-, Visual Basic-, ja .NET-ympäristöille. Vision Acquisition Software on ajuri- ja työkalupaketti eri tyyppisille konenäkökameroille ja sisältyy sekä Vision Builderiin että Vision Development Moduleen. Paketti sisältää NI-IMAQ-, NI-IMAQdx- ja NI-IMAQ I/O-ajurit, joista kukin pitää sisällään myös joukon LabVIEW VI:ta. [23, 24.] IMAQ:in VI:ta hyödynnettiin testauslaitteen ohjelmistossa erilaisiin kuvankäsittely- ja analyysitehtäviin.

Vision Development Modulen mukana tulee Vision Assistant -työkalu, jonka avulla käyttäjä voi kehittää ja testata kuvankäsittely ja -analyysialgoritmeja menupohjaisessa käyttöliittymässä. Luotu algoritmi on mahdollista kääntää LabVIEW- tai C-kielelle. [25 s. 9–10.] Tässä työssä Vision Assistantia käytettiin kuvista viivoja etsivän algoritmin luomiseen. Vision Assistantilla tehty algoritmi käännettiin LabVIEW-muotoon ja yhdistettiin osaksi laajempaa LabVIEW-ohjelmaa (Klusterianalyysi, osio 4.4).

Osana insinöörityötä oli selvittää NI Visionin hyödyllisyyttä tulevia kuvaprosessointia vaativia projekteja ajatellen.

3.2.1 Reunojen haku NI Visionissa

NI Visionissa reuna on määritelty merkittävänä harmaansävyarvon muutoksena vierekkäisten pikseleiden välillä. Käyttäjä voi määritellä reunanhakualueen, sen muodon sekä reunan minimivahvuuden eli pienimmän merkittävän eron taustan ja reunan harmaasävyarvoissa Minimum Edge Strength tai Treshold Level -parametrin avulla. Reunanhaku voidaan tehdä käyttäen yksiulotteista (1D) tai kaksiulotteista (2D) hakualuetta. 2D-hakualue koostuu useista hakulinjoista, joiden lukumäärän ja haun suunnan käyttäjä voi määritellä. [26.]

2D-reunanhakuun NI Visionissa on kolme työkalua:

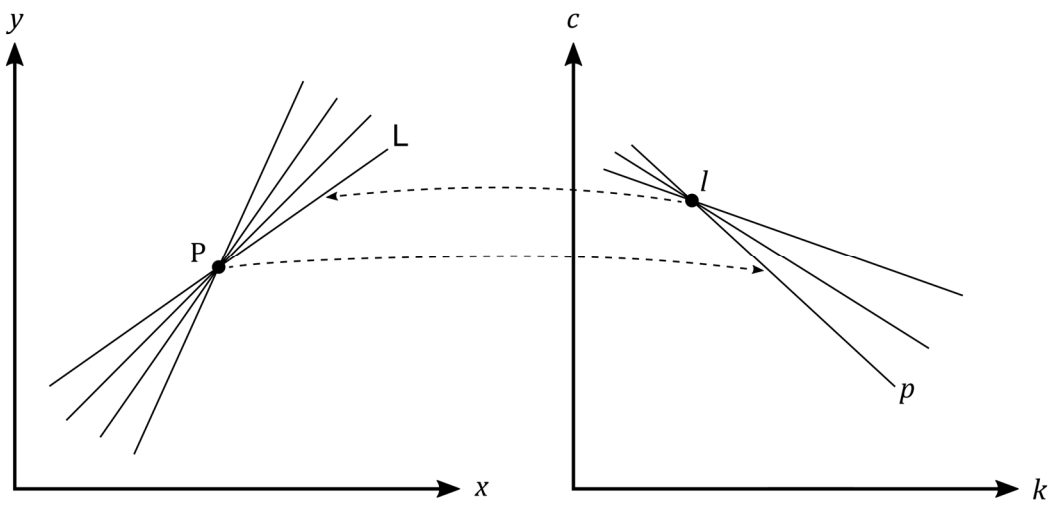
- Rake-työkalussa hakualue on suorakulmio ja reunanhakulinjat samansuuntaisia suorakulmion kanssa.
- Spoke-työkalu käyttää ympyränmuotoista hakualuetta ja hakulinjat kulkevat ympyrän keskeltä sen reunoille.
- Concentric rake -työkalussa hakualue on ympyränmuotoinen ja hakulinjat ovat ulkoreunan kanssa samankeskisiä ympyröitä.

Suorien reunojen hakuun NI Visionissa on useita eri menetelmiä, joista tässä työssä käytettiin Houghin muunnokseen perustuvaa Hough-pohjaista reunanhakua [26].

3.2.2 Houghin muunnos

Houghin muunnos on menetelmä suorien ja muiden geometristen muotojen hakemiseen kuvasta. Ennen Houghin muunnosta tutkittava kuva siivotaan ylimääräisistä yksityiskohdista suodatuksen, kynnystyksen tai reunantunnistuksen avulla tai näiden yhdistelmällä. Tässä opinnäytetyössä tämä tehtiin muuttamalla kuva binäärimuotoon ja kynnystämällä arvoja. Houghin muunnos löytää myös epätäydelliset muodot, esimerkiksi katkonaiset suorat. Muunnoksessa kuitenkin katoaa informaatiota: esimerkiksi suorien tai kaarien päätepisteet on määritettävä jälkeenpäin. Houghin muunnosta on käytetty suorien lisäksi muun muassa ympyröiden ja ellipsien detektioon. Haettavan muodon määrittävien parametrien määrän kasvaessa myös tietokoneen resursseihin kohdistuvat vaatimukset kasvavat, mikä tekee menetelmästä raskaan monimutkaisemmille muodoille. Hough on kuitenkin mahdollista yleistää mille tahansa muodolle. [27.]

Houghin muunnoksessa suorien tunnistus kuvasta pohjautuu suoran ja pisteen väliseen kaksinaisuuteen (kuva 9): suora voidaan määritellä sillä sijaitsevien pisteiden ja piste sen läpi kulkevien suorien avulla. Kuvassa (x,y-avaruus) oleva suora $y = kx + c$ voidaan esittää parametriavaruudessa (k,c-avaruus) pisteenä. Houghin muunnoksessa kuvan pisteet siirretään x,y-avaruudesta k,c-parametriavaruuteen, missä kutakin pistettä vastaa suora. Suoralla L sijaitsevat pisteet määrittelevät parametriavaruudessa joukon suoria, jotka leikkaavat toisensa pisteessä *I*. Pisteen *I* koordinaatit ovat viivan L parametrit (k ja c).



Kuva 9. Houghin muunnos. Mukaillen [27].

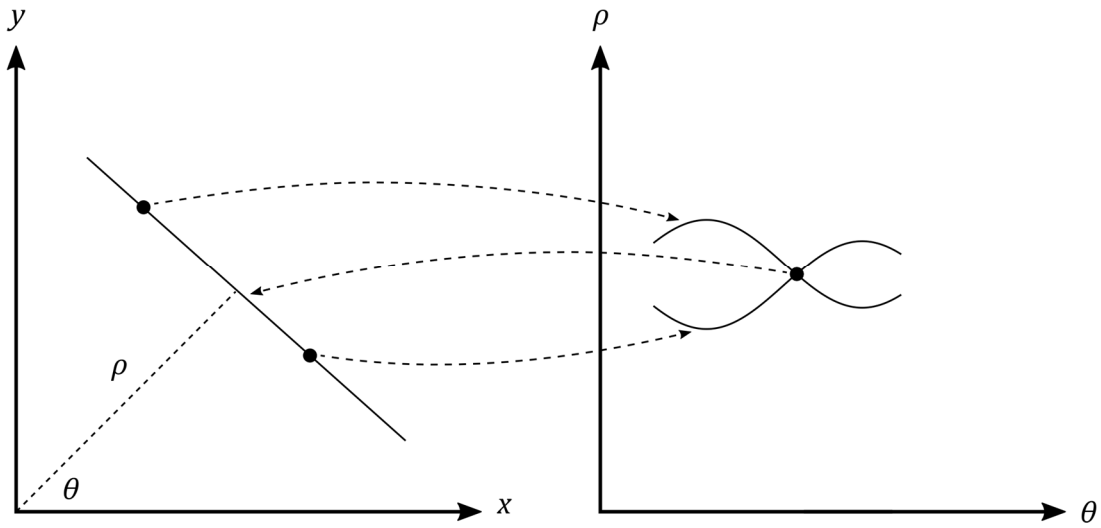
Tutkitaan kuvassa (*x*,*y*-avaruus) olevaa pikseleistä koostuvaa suoraa joka voi olla paikoin katkonainen. Tutkittavalla suoralla L on vastaava piste *l* *k*,*c*-avaruudessa (parametriavaruus), joka jaetaan pieniin soluihin. Jokaista kuvan pikseliä vastaa parametriavaruudessa suora, jonka kulku kunkin solun läpi tallennetaan. Pistettä *l* vastaava solu saa paljon osumia, minkä avulla kuvassa oleva suora tunnistetaan. Ei-lineaariset kohteet aiheuttavat taustaosumia, minkä vuoksi oikean kynnysarvon valitseminen on tärkeää. Jos kynnysarvo on liian korkea, vain muutamasta pikselistä koostuva viiva voi hukkua taustaan. [27.]

Yhtälön $y = kx + c$ ongelmana ovat vertikaaliset suorat, joiden kulmakerroin on määrittelemätön. Houghin muunnoksesta käytetäänkin yleisemmin Dudan ja Hartin kehittämää muotoa

$\rho = x \cos \Theta + y \sin \Theta$, missä

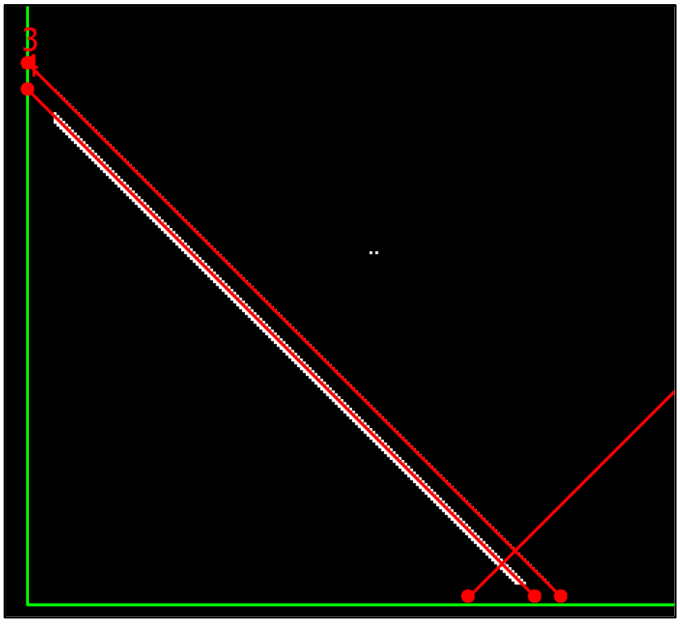
- ρ on suoran etäisyys origosta
- Θ on suoran normaalin kulma.

Tätä muotoa käytettäessä kuvan pisteet määrittävät parametriavaruudessa suorien sijaan sinikäyriä (kuva 10). Suora *x*,*y*-avaruudessa vastaa sinikäyrien leikkauspistettä *ρ*,*Θ*-avaruudessa, mikä näkyy kuvassa 10 tyypillisenä ”perhoskuviona”.



Kuva 10. Houghin muunnos Dudan ja Hartin mukaisesti.

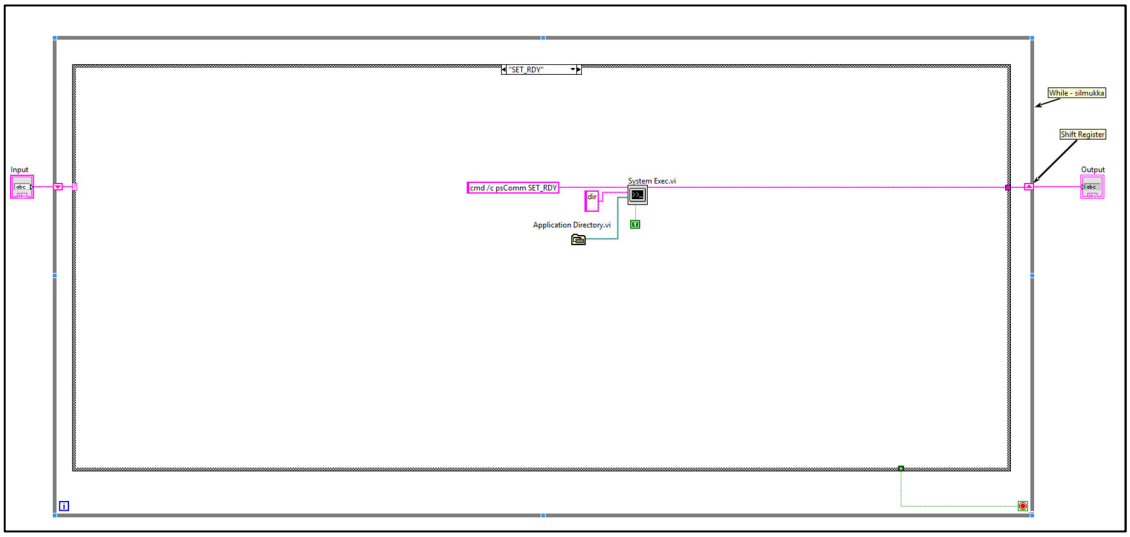
Houghin muunnoksen toteuttamisessa hyödynnettiin tässä työssä NI Visionia, joka käyttää suoran parametrisointiin yhtälöä ρ = x cos Θ + y sin Θ [26]. Ennen Houghin muunnosta käsiteltävä kuva muutettiin binäärimuotoon Extract Luminance Plane -funktiolla ja kynnystettiin Threshold-funktiolla. Kuvassa 11 on esimerkki kuvasta, josta NI Vision on tunnistanut viivoja Houghin muunnoksen avulla.



Kuva 11. Houghin muunnos NI Visionissa. Valkoiset viivat muodostuvat huonoista pikseleistä, punaiset viivat ovat Hough Edge Rake-funktiolla löydettyjä ja merkittyjä viivoja. Kaksi kolmesta kuvassa olevista viivoista ovat yhden pikselin paksuisia, joten ne peittyvät punaisten viivojen alle.

4 Tulokset

4.1 psComm-kommunikaattori



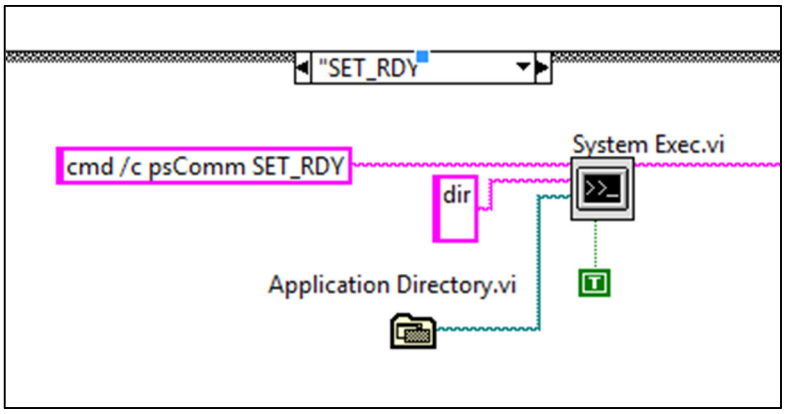
Kuva 12. psComm-kommunikaattori SET_RDY-tilassa. Kuvassa näkyy ohjelman state machine -rakenne.

Insinöörityöprojektin alussa kehitettiin yksinkertainen LabVIEW-ohjelma, joka ottaa vastaan käyttäjän käskyjä ja muuttaa ne sopivaan muotoon Planmecan psComm-ohjelmaa varten. psComm on apuohjelma, joka kommunikoi ProSensor USB CDC:n (*engl.* communications device class) kanssa. Ohjelma myös muuttaa sensorien testauslaitteen psCommin kautta lähettämät vastaukset ymmärrettävään muotoon. Ohjelma toimii osana laajempaa LabVIEW-ohjelmistoa.

psComm-ohjelma sisältää XXX_XXX-muotoisia käskyjä. Joidenkin käskyjen perään lisätään välilyönnin jälkeen parametreja, esimerkiksi sensorin uusi sarjanumero. psComm-kommunikaattori hyödyntää state machine -arkkitehtuuria (osio 3.1.1), jossa while-silmukka pyörittää sisällään olevaa case-rakennetta. Jokaisella käskyllä on samanniminen tila, joka aktivoituu ja suorittaa sisältämänsä koodin kun käyttäjä syöttää nimeä vastaavan käskyn ohjelmaan (kuva 13). Ohjelman sisältämät tilat on kuvattu tarkemmin alla.

Ohjelma lähettää käskyn psComm-ohjelmalle aina samalla rakenteella (kuva 13). Ohjelma avaa psComm-komentoikkunan ja syöttää tilan nimeä vastaavan käskyn sekä mahdolliset parametrit System Exec.vi -funktioon, joka käynnistää tai suorittaa

Windows-pohjaisia ohjelmia tai komentoriviohjelmia. Loppukäyttäjän tietokoneen hakemistolla voi olla eri nimi kuin testauksessa käytetyllä tietokoneella. psComm-ohjelma sijoitetaan samaan hakemistoon kuin psComm-kommunikaattori. System Exec.vi'hin kytketään Application Direcory.vi-noodi, joka palauttaa polun ohjelman ja psCommin sisältävään hakemistoon. Näin System Exec.vi saa aina oikean polun psComm-ohjelmalle.



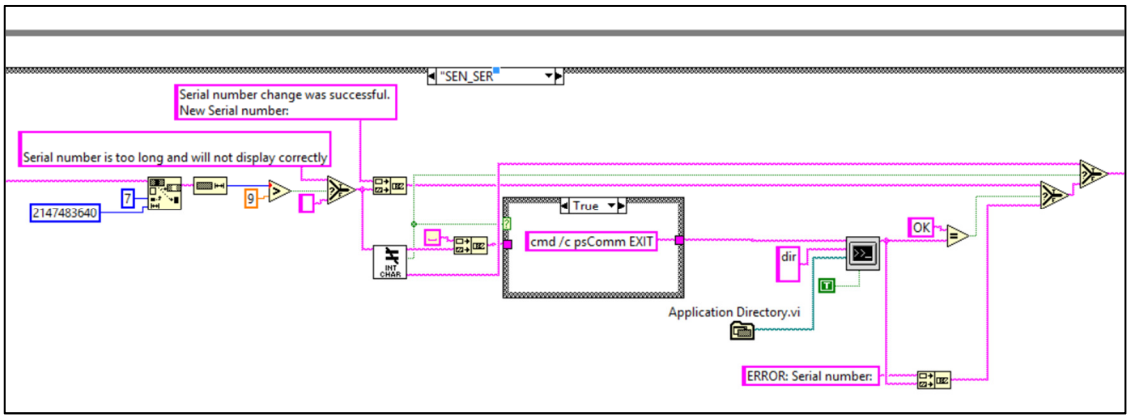
Kuva 13. SET_RDY-tila. Tilassa sensori asetetaan valmiustilaan odottamaan käskyä käyttäjältä.

Ohjelmassa on kolme komentoa, SET_RDY (Set Ready), BRK_IMG (Break Imaging) ja SEN_PWR (Sensor Power), jotka kontrolloivat sensorin tilaa. SET_RDY-komento (kuva 13) asettaa sensorin valmiustilaan, BRK_IMG lopettaa mahdollisesti käynnissä olevan kuvantamisen ja SEN_PWR kontrolloi sensorin virtatilaa. SEN_PWR 1 asettaa sensorin ON-tilaan ja SEN_PWR 0 OFF-tilaan. Mikäli System Exec.vi ei tulosta joko 1 tai 0, case-rakenne aktivoi vakiotilan ”ERROR”, joka tulostaa tekstin ”Error:” ja tämän perään System Exec.vi'n tulosteen.

Osa ohjelman komennoista (GET_STS, SEN_SER, GET_PAR, GET_PAG, PRG_PAG, FPG_VRS, GET_VRS) palauttaa tietoja sensorista tai ohjausyksiköstä tai tekee niihin muutoksia. GET_STS (Get Status) palauttaa kontrolliboxin valotustilan. System Exec.vi tulostaa numeroita, jotka merkitsevät eri valotustiloja. Numeron ollessa kolme tai suurempi kontrolliboksi on valmis vastaanottamaan käskyjä.

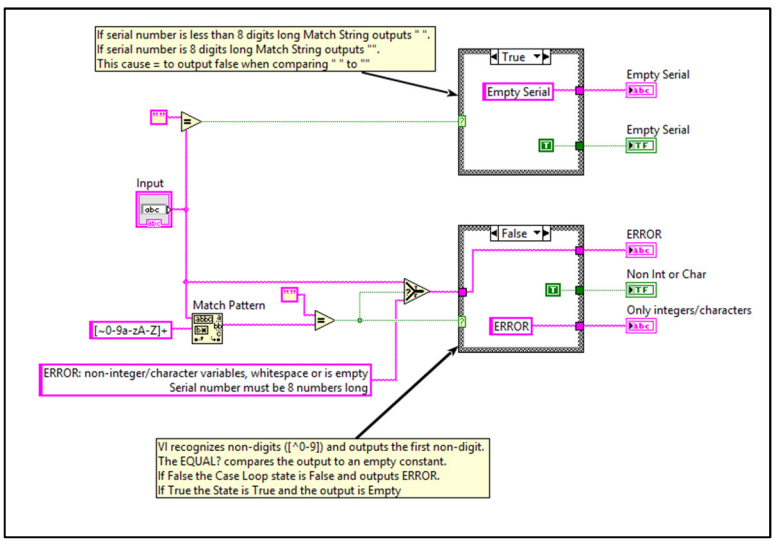
SEN_SER-komento (Sensor Serial) palauttaa tai kirjoittaa sensorin sarjanumeron (kuva 14). Ilman parametria annettuna komento palauttaa sarjanumeron. Mikäli komento annetaan parametrin kanssa, esimerkiksi SEN_SER 1218567, annettu lukujono

kirjoitetaan sensorin sarjanumeroksi ja ohjelma palauttaa ”OK”. psComm-ohjelma ei tarkista käyttäjän antamaa sarjanumeroa. Jotta vääränmuotoisen sarjanumeron syötöltä vältyttäisiin, psComm-kommunikaattoriin lisättiin subVI, joka tarkistaa käyttäjän syöttämän merkkisarjan koostumuksen (kuva 15). Sarjanumeron maksimipituus on seitsemän merkkiä ja se saa sisältää ainoastaan numeroita ja kirjaimia. SubVI ilmoittaa myös, mikäli syöte on tyhjä. Mikäli syöte on virheellinen, ohjelma palauttaa ”ERROR”.

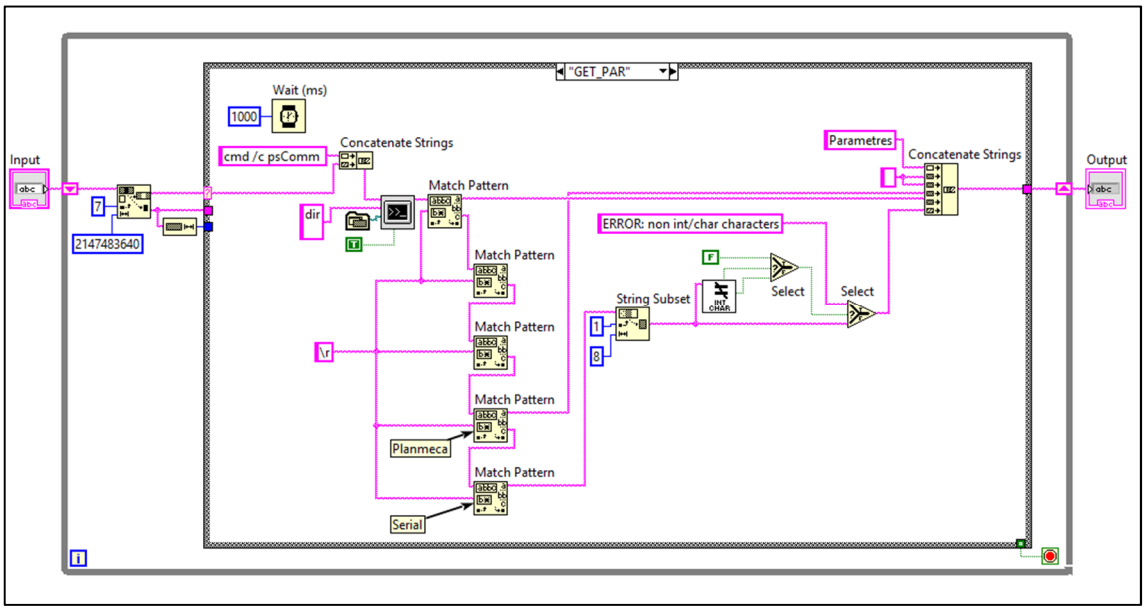


Kuva 14. SEN_SER-tila. Sarjanumero tarkistetaan ennen syöttöä psComm-ohjelmaan virhetilanteiden välttämiseksi.

GET_PAR-komento (Get Parameters) tulostaa kuvan koon, leveyden, korkeuden ja sarjanumeron (kuva 16). Ohjelma erottaa parametrit, tarkistaa sarjanumeron ja yhdistää parametrit ennen tulostusta. Jos System Exe.vi palauttaa virheen ohjelma palauttaa ”ERROR”.



Kuva 15. ”Not an int or char”-subVI tutkii, onko käyttäjän syöttö ohjelmaan hyväksyttävä.

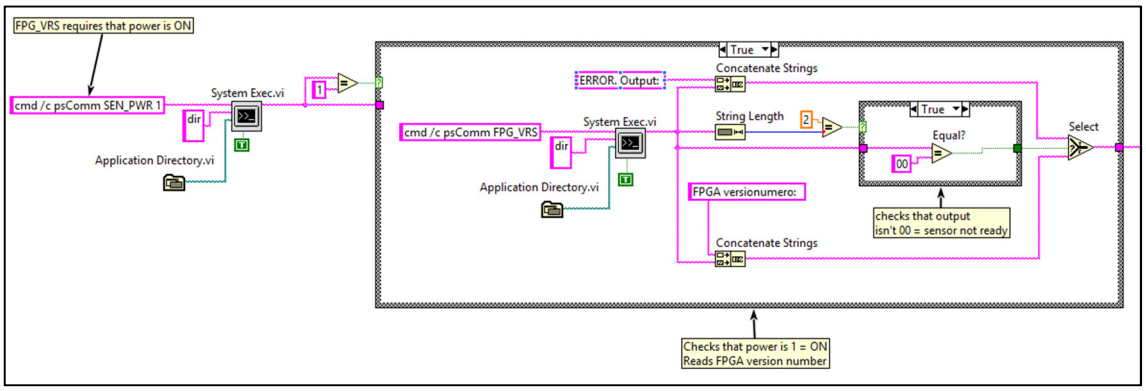


Kuva 16. GET_PAR-tila. psComm-kommunikaattori pyytää sensorilta parametrit, tarkistaa psComm-ohjelman tulosteen ja muuttaa sen ymmärrettävään muotoon.

GET_PAG palauttaa sensorin FPGA-sivulle (engl. *Field Programmable Gate Array*, uudellen ohjelmoitava digitaalinen mikropiiri) 0, 1 tai 2 ohjelmoidun materiaalin. Palautettu sivu tallennetaan annettuun tiedostoon. Käsky on muotoa ”GET_PAG sivun numero polku\tiedostonimi.bin”. Virhetilanteessa tila tulostaa ”unable to execute GET_PAG” ja sensorin virhetulostuksen. Toimiessaan ohjelma tulostaa System Exec.vi -tulosteen, ”GET_PAG was successful.” sekä tallennustiedoston nimen ja sijainnin.

PRG_PAG ohjelmoi valitun tiedoston sisällön sensorin FPGA:n sivulle 1 tai 2. Komento annetaan muodossa ”PRG_PAG sivunumero polku\tiedostonimi.bin”. Tila toimii virhetilanteessa samoin kuin GET_PAG.

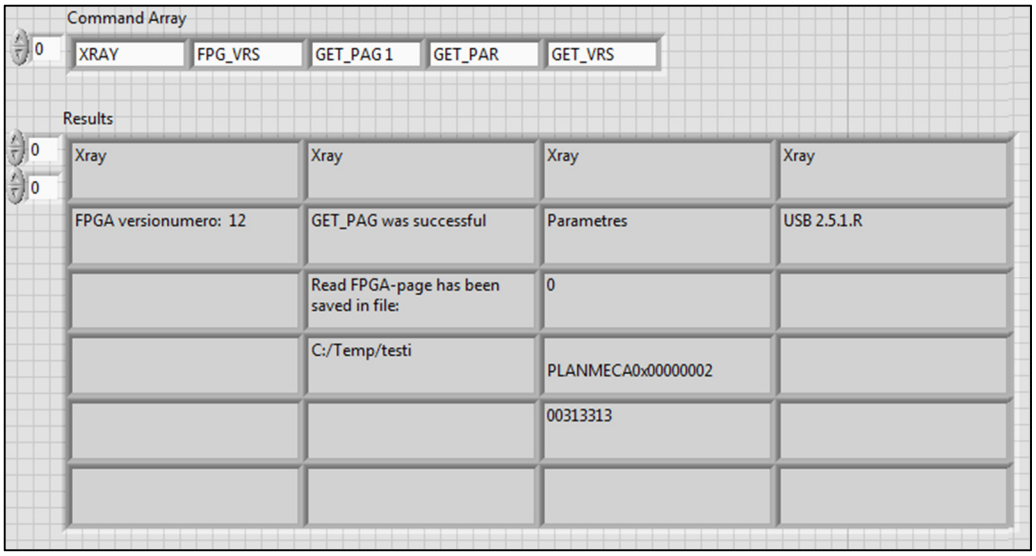
FPG_VRS (kuva 17) palauttaa FPGA:n versionumeron. Jotta FPGA:n versionumero voidaan lukea, sensorin on oltava päällä. Tämän varmistamiseksi ohjelma lähettää sensorille SEN_PWR 1 -komennon. Ohjelma tulostaa ”1”, jos sensori on päällä tai virheilmoituksen. Jos paneeli on päällä, ohjelma lähettää FPG_VRS -komennon. Ohjelma tarkistaa versionumeron pituuden ennen sen tulostamista. Numeron ollessa kaksi merkkiä pitkä case-rakenne tarkistaa oliko tuloste ”00”, mikä merkitsee, että sensori ei ollut valmis; tällöin ohjelma tulostaa virheilmoituksen. Mikäli tuloste on pituudeltaan erisuuri kuin 2, ohjelma tulostaa suoraan FPGA-versionumeron.



Kuva 17. FPG_VRS -tila. Ohjelma tarkistaa, että sensori on valmiustilassa. Tämä jälkeen ohjelma lähettää käskyn psComm-ohjelmalle, joka palauttaa FPGA-versionumeron.

GET_VRS-käsky palauttaa sensorin versionumeron. Toimiessaan System Exec.vi tulostaa ”USB” ja versionumeron. Ohjelma vertaa tulosteen kolmea ensimmäistä kirjainta USB-kirjainjonoon. Jos tuloste on ”USB”, ohjelma tulostaa ohjelmiston versionumeron. Muutoin ohjelma tulostaa ”ERROR: ” ja virhetulosteen.

4.2 Sarjanumeron luku -VI



Kuva 18. Sarjanumeron luku -VI:n etupaneeli.

Testauslaitteella suoritetaan erilaisia toimintoja, esimerkkinä kalibrointi, joissa käyttäjä tarvitsee laitteelta eri parametreja. Sarjanumeron luku -VI:ssa käyttäjä syöttää valitsemansa päätilat matriisiin ja käynnistää ohjelman. Ohjelma koostuu kahdesta case-raken-

teesta, joista ulompi valitsee päätilan ja sisempi komentotilan. Päätilat vastaavat ulomman case-rakenteen tiloja. Calibration-, Xray- ja Repair-päätilojen sisällä on pienempi case-rakenne joka koostuu komentotiloista. Pienempi case-rakenne valitsee Command Array -matriisin perusteella toteutettavan komentotilan, jossa komento suoritetaan. Näin käyttäjä voi halutessaan jättää pois käskyjä tai muuttaa niiden järjestystä. Komentotilat sisältävät kutakin komentoa vastaavan suoritettavan ohjelmakoodin.

Ohjelma rakennettiin psComm-ohjelman pohjalta ja komentotilat, joissa toiminnot tapahtuvat, ovat pitkälti samoja kuin psComm-ohjelmassa. Ohjelmassa on seuraavat päätilat:

- LED-tilassa ohjelma pyytää käyttäjää syöttämään sensorin siru-ID:n. Ohjelma tulostaa ”LED” sekä syötetyn merkkijonon.
- Calibration-tila tulostaa kuvan parametrit, ohjelmiston USB-numeron, FPGA-version ja tallentaa luetun FPGA-version käyttäjän valitsemaan tiedostoon. Calibration-tilassa on mahdollista, että sensorille ei ole asetettu sarjanumeroa, jolloin se täytyy asettaa. Ohjelma pyytää tällöin käyttäjää syöttämään sarjanumeron.
- Repair- ja Xray -tiloissa komentotilat ovat samat kuin Calibration-tiloissa. Näissä tiloissa ei ole tyhjän sarjanumeron tarkistusta, sillä sensorilla voidaan olettaa olevan sarjanumero.
- Virheitä varten on ERROR-virhetila.

Käyttäjä kirjoittaa etupaneelissa (kuva 18) olevaan Commad Array -matriisiin haluamansa tilan ja käskyt. Alla olevaan Results-matriisin soluihin tulee tilan nimi sekä ohjelman komentokohtainen tuloste. Virhetilassa tilaa vastaavaan soluun tulee ”Error:” sekä virheellinen tuloste tai mahdollinen virheen aiheuttaja.

Ohjelman käynnistyessä sensori asetetaan päälle ja mahdollisesti käynnissä oleva kuvantaminen lopetetaan.

Ohjelma sisältää kolme subVI:ta:

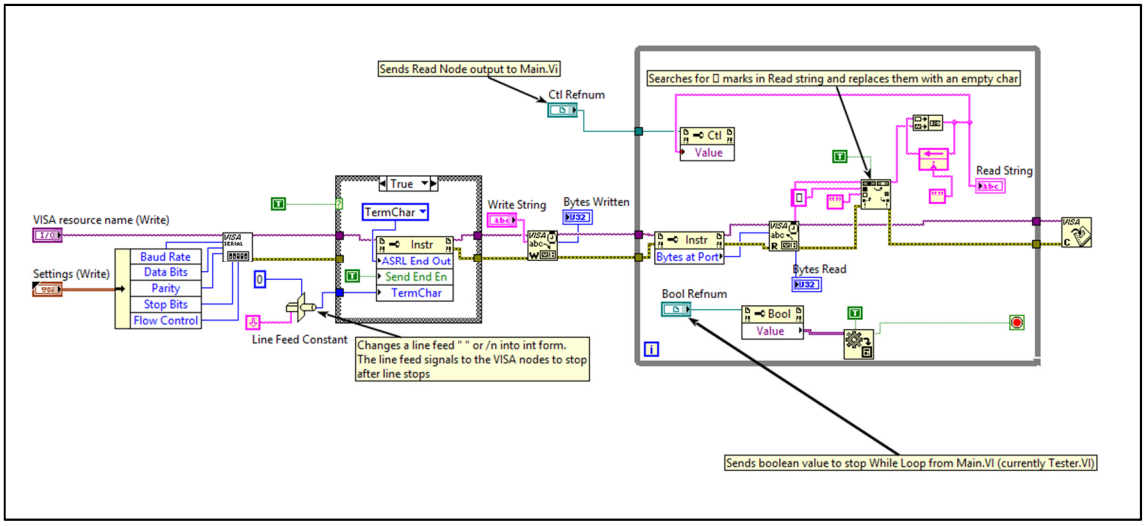
- Check Sensor Ready -subVI pyörittää BRK_IMG, SET_RDY, GET_STS -käskysekvenssiä, joka valmistelee ja tarkistaa sensorin statuksen aina valitun päätilan käynnistyessä. VI lähettää sarjan käskyjä psComm-ohjelmalle ja tulosteen perusteella päästää ohjelman etenemään sisempään case-rakenteeseen tai virhetilaan.
- Not an INT/Char -subVI tarkistaa sisältääkö käyttäjän syöttämä sarjanumero virheellisiä merkkejä. Jos sarjanumero sisältää muita merkkejä kuin

numeroita tai kirjaimia, ohjelma antaa virheilmoituksen. Jos sensorilla ei ole sarjanumeroa Calibration-päätilassa, Not and INT/CHAR -subVI kertoo Empty Serialnumber -subVI:lle ongelmasta. Muissa päätiloissa tilaa vastaavan tulosmatriisin soluun tulostuu sarjanumeron sijaan virheilmoitus.

- Empty Serialnumber -subVI (vain Calibration-tilassa) pyytää käyttäjää asettamaan sarjanumeron mikäli sensorilla ei ole sarjanumeroa. Mikäli sarjanumero on hyväksyttävä, ohjelma jatkaa normaalisti. Virheellisen tai tyhjän sarjanumeron tapauksessa Empty Serialnumber asettaa sensorin oikeaan tilaan sarjanumeron kirjoittamista varten ja pyytää käyttäjää antamaan uuden sarjanumeron.

Ohjelman tulosteet kerätään ohjelman alussa luotuun matriisiin. Lopuksi kerätyt tulosteet tulostetaan etupaneelissa sijaitsevaan Results-matriisiin.

4.3 RS-232-kommunikointi



Kuva 19. RS-232-kommunikointiohjelman lohkokaavio.

Seuraavaksi rakennettiin ohjelma testauslaitteen ja tietokoneen RS-232-sarjaporttien väliseen kommunikointiin. Apuna käytettiin LabVIEW'n RS-232-kommunikointimallia [28].

Sarjaliikenteessä portti vastaanottaa ja lähettää dataa bitti kerrallaan. Lähetyksissä kulkeva data on yleensä ASCII-tyyppistä ja voi sisältää 5, 6, 7 tai 8 databittiä. Kommunikointi tapahtuu kolmen linjan kautta, jotka ovat lähetys, vastaanotto ja maa. [29.] Synkronisessa sarjaliikenteessä laitteet tahdistavat itsensä toistensa suhteen. Tämän jälkeen tapahtuva kommunikointi on jatkuvaa: kun varsinaista dataa ei lähetetä, laitteet pysyvät tahdissa lähettämällä erityistä synkronisointimerkkiä (SYN, engl. *synchronous idle*).

Asynkronisessa sarjaliikenteessä ei käytetä synkronointimerkkiä, vaan jokaisen lähetettävän tavun alkua ja loppua merkitsevät ylimääräinen alku- ja loppubitti. [30.]

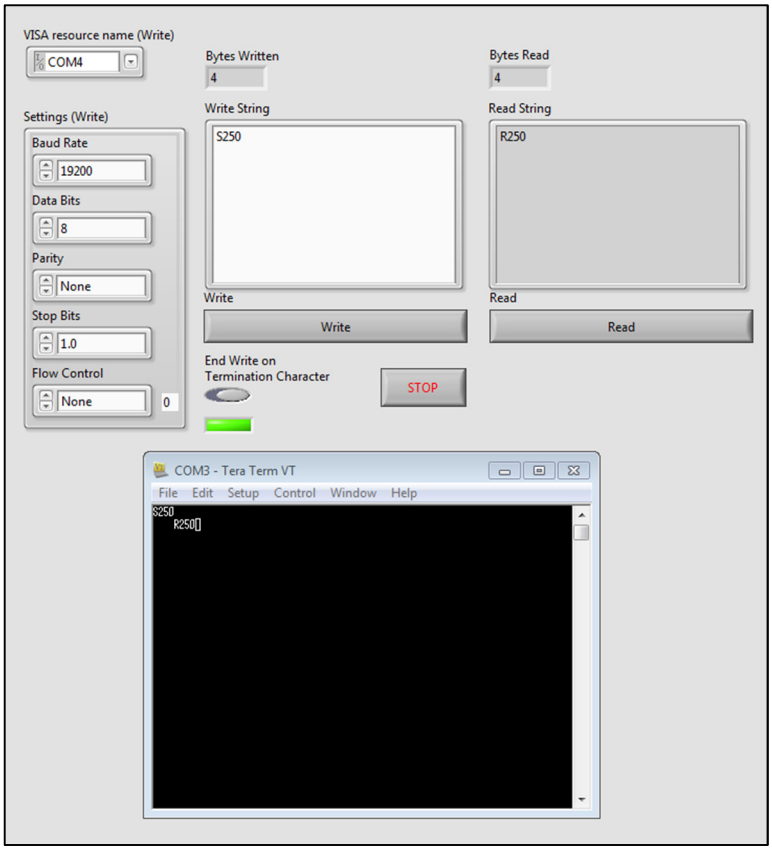
Jotta kommunikaatio porttien välillä olisi mahdollista on seuraavien parametrien oltava samat porttien välillä [31]:

- Baud Rate kuvaa tiedonsiirron nopeutta. Baudi-yksikkö vastaa yleensä bittejä sekunnissa (bps) eli 1 baudi = 1 bps.
- Data Bits ilmoittaa databittien määrän yksittäisessä paketissa. Riippuen lähetetyn datan tyypistä databittien määrä voi olla 5–8. ASCII-merkistö on 7-bittinen, mutta erikoismerkkien ilmaisemiseksi voidaan käyttää jotain laajennetuista 8-bittisistä merkistöistä.
- Stop Bits kuvaa loppubittien kestoa, joka voi olla 1, 1.5 tai 2. Loppubitit ilmoittavat yksittäisen datapaketin päättymisestä.
- Parity: Paritettibitin avulla voidaan havaita, onko tiedonsiirrossa tapahtunut virhe. Mahdolliset asetukset ovat None, Even, Odd, Space ja Mark. Jos asetus on None, pariteettibittiä ei käytetä. Jos käytetään asetusta Even tai Odd, lähettävä laite tarkistaa databiteissä olevien ykkösten määrän ja asettaa pariteettibitin arvoksi joko 1 tai 0 niin, että sanassa (databitit + pariteettibitti) olevien ykkösten määrä on joko parillinen (Even) tai pariton (Odd). Vastaanottava laite tarkistaa vastaanotetun datan. Mikäli datassa on sanoja jotka poikkeavat valitusta pariteetista, tiedonsiirrossa on tapahtunut virhe. Mikäli tapahtuu useita virheitä siten, että sanan pariteetti ei muutu, virhettä ei havaita. Mark ja Space ovat harvoin käytettyjä asetuksia, joissa pariteettibitin arvon on aina joko 0 (Space) tai 1 (Mark). [32.] Paritya ei käytetty testausvaiheessa.

Flow Control on tapa kontrolloida porttien välistä kommunikaatiota, jolla voidaan varmistaa että vastaanottava laite ei ylikuormitu tulevan datan määrästä. Mahdolliset asetukset ovat None, XON/XOFF, RTS/CTS ja DTR/DSR. XON/XOFF on ohjelmistopohjainen flow control -menetelmä (engl. *software flow control*), kun taas RTS/CTS ja DTR/DSR ovat laitteistopohjaisia flow control -menetelmiä (engl. *hardware flow control*). [33, 34.] Testauksessa ei käytetty Flow Controlia.

Kuvassa 19 on esitetty RS-232-kommunikointiohjelman lohkokaavio ja kuvassa 20 etupaneeli. Etupaneelissa parametrit ovat sijoitettuna klusteriin. Visa resource name (Write) -kohdasta valitaan kanava, jonka kautta ohjelma lähettää esimerkiksi käskyjä testauslaitteelle. End Write on Termination Character -valitsimesta käyttäjä voi käskeä sarjaportin lopettamaan kirjoittamisen portin havaitessa lopetusmerkin. Lopetusmerkki kertoo lähetyksen päättyneen. Lopetusmerkkiä ei yleensä vaadita sillä kommunikaatioprotokollan implementaatiolla on yleensä oma metodi kertoa lähetyksen päättyneen. Ohjelmassa

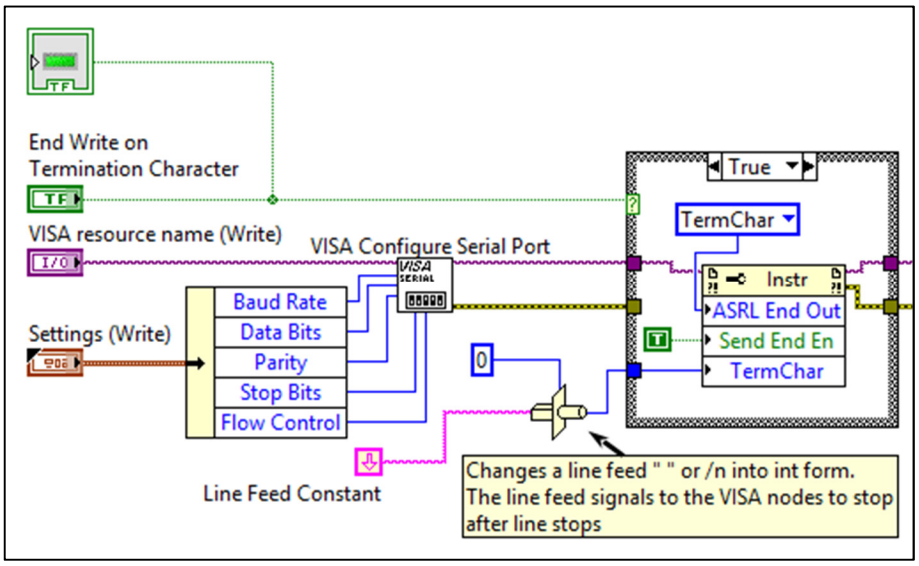
lopetusmerkin arvo on 0, jolloin lopetusmerkkiä ei lisätä, mutta ohjelmaan voi helposti lisätä lopetusmerkin. [35.]



Kuva 20. RS-232-kommunikointiohjelman etupaneeli.

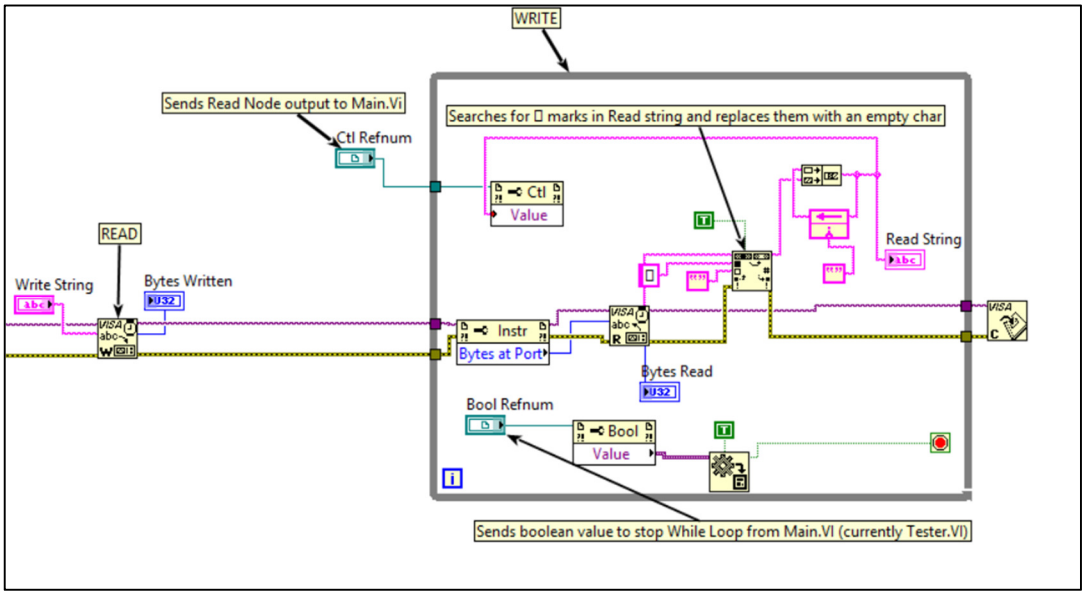
Kommunikoinnin valvomiseksi etupaneelissa on lähetettyjen ja luettujen tavujen määrä (Bytes Written ja Bytes Read). Write String -kontrolliin kirjoitetaan haluttu teksti. Yhteyden toimiessa sama teksti ilmestyy Read String -indikaattoriin. Ohjelman toimintaa testattiin Tera Term -terminaaliemulaattorilla (kuva 20) yhdistämällä kaksi sarjaporttia toisiinsa, asettamalla parametrit samoiksi ja valitsemalla portteja vastaavat kanavat.

Ohjelmassa käytetään VISA-noodia, joka valmistelee portin annetuilla parametreilla [28]. Kuvassa 21 vasemmalla olevat kontrollit sisältävät käyttäjän asettamat parametrit. Asetukset ovat tallennettuna klusteriin, josta ne puretaan ja syötetään VISA Configure Serial Port -noodiin. Jos käyttäjä asettaa lopetusmerkin päälle, case-rakenteen True-tilan sisällä oleva property-noodi muokkaa VISA Configure Serial Port -ominaisuuksia niin että lähetys loppuu lopetuskirjaimeen. Muussa tapauksessa data kulkee case-rakenteen läpi muuttumattomana.



Kuva 21. RS-232-kommunikointiohjelman alkuosa.

Kun käyttäjä haluaa lähettää etupaneelille kirjoittamansa tekstin hän painaa Write-painiketta. Tällöin painiketta lohkokaaviossa vastaava TF-kontrolli tulostaa True-arvon, joka asettaa case-rakenteen kirjoitustilaan ja Visa Write -noodi lähettää käyttäjän tekstin laitteelle (kuva 22).



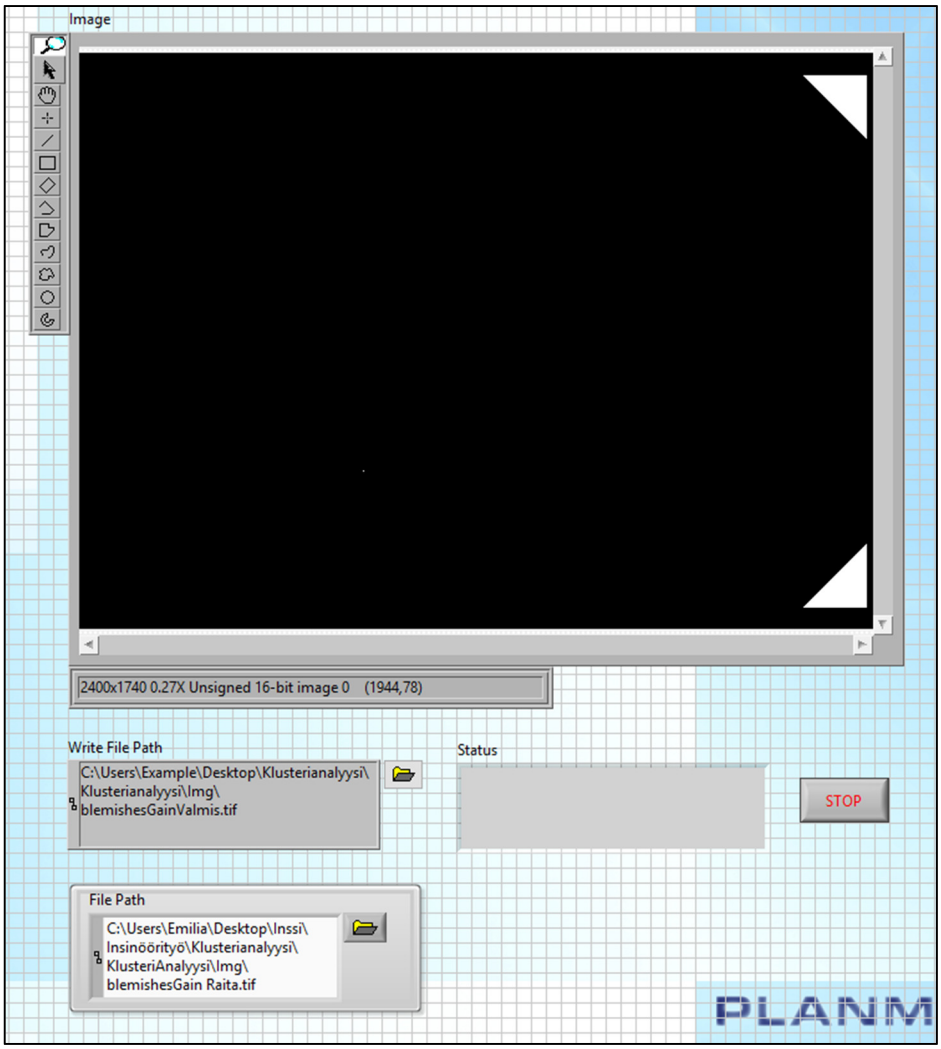
Kuva 22. RS-232-kommunikointiohjelman loppuosa.

Käyttäjän painaessa etupaneelin Read-painiketta case-rakenne siirtyy True-tilaan. Property node hakee portilla olevien bittien määrän ja lähettää tiedon Visa Read -noodille, joka lukee annetun määrän verran bittejä (kuva 22). Search and replace -noodi hakee

tulosteesta ””-merkkejä ja korvaa ne tyhjällä. Muutoin ohjelma jatkaa kunnes käyttäjä painaa Read- tai Stop-painiketta, jolloin VISA Close sulkee yhteyden porttien välillä.

4.4 Klusterianalyysi

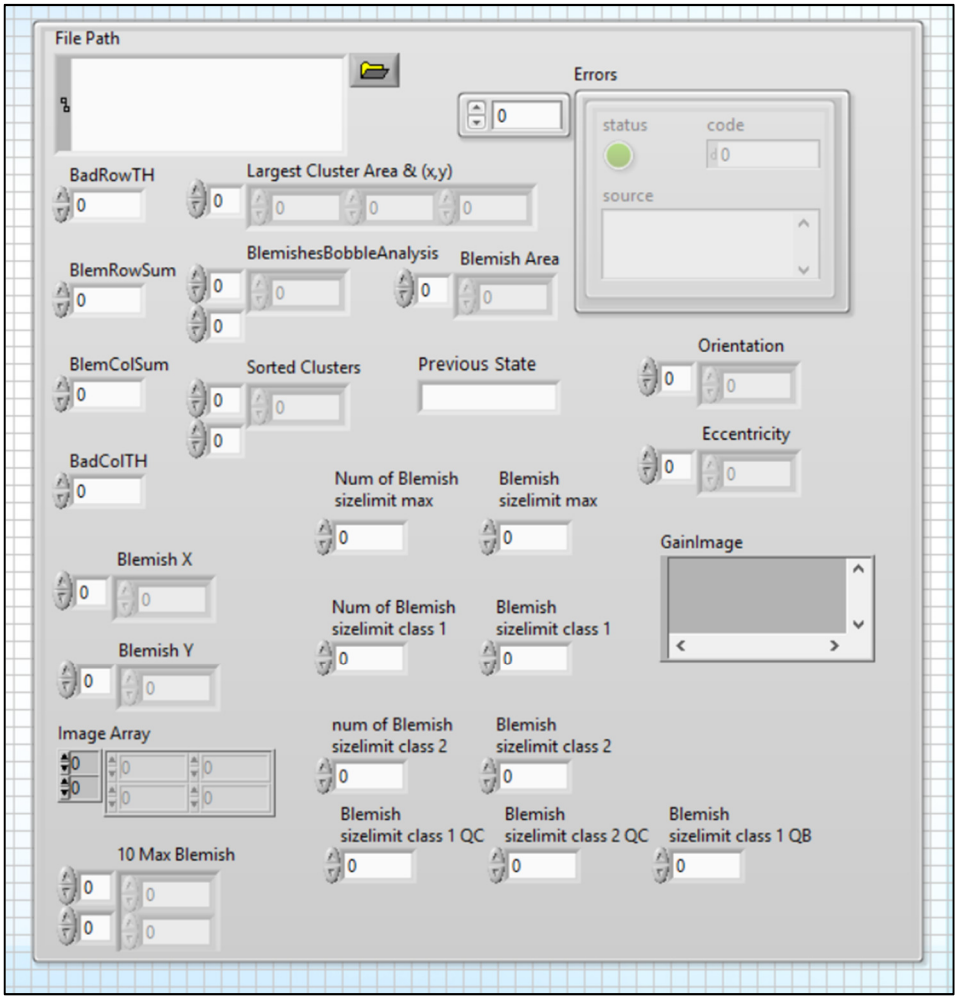
Klusterianalyysi on yksi joukosta analyyseja, jotka suoritetaan osana intraoraalisensorin testausta. Klusterianalyysi keskittyy viallisten pikseleiden hakuun, määrittelyyn ja muokkaamiseen testikuvassa. Lopuksi ohjelma päättää tulosten perusteella, läpäiseekö sensori raja-arvot vai ei.



Kuva 23. Klusterianalyysin etupaneeli.

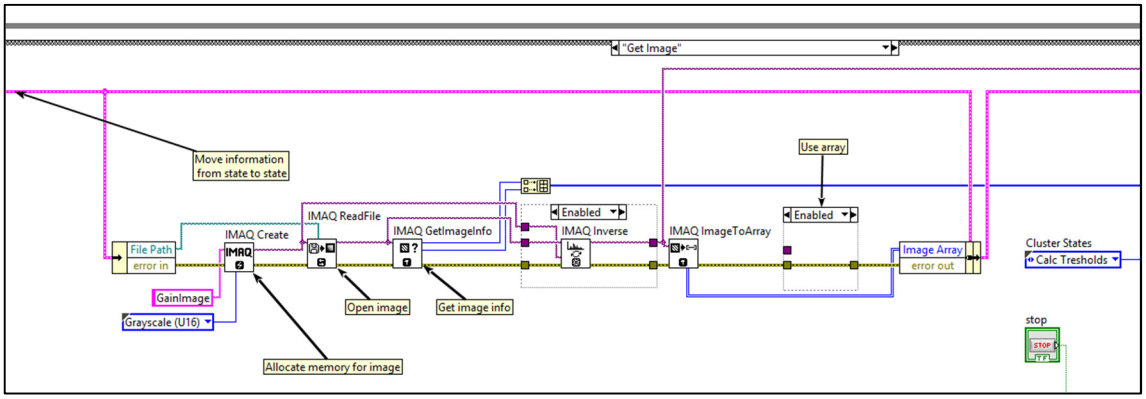
Intraoraalisensorin testauskuville tehtävät analyysit oli aiemmin toteutettu MATLAB-ohjelmana, mutta testaussekvenssi haluttiin kääntää LabVIEW'lle. Alla olevassa vertaillaan Klusterianalyysin MATLAB- ja LabVIEW-toteutuksia.

Kuvassa 23 on LabVIEW-ohjelman etupaneeli. Etupaneeli ei ole lopullinen, lisättäessä pääohjelmaan siihen tulee todennäköisesti muutoksia. Etupaneelissa on ruutu, josta seuratiin kuvan käsittelyä. File Path -kontrollilla haetaan käsiteltävä kuva. Write File Path -kontrolliin syötetään polku käsitellyn kuvan tallentamiselle sekä tiedoston nimi. Status-laatikko kertoo analyysin lopuksi oliko kuvassa liikaa virheitä. Tarvittava informaatio kuten raja-arvot tallennetaan Infoklusteriin (kuva 24), jossa informaatio kuljetetaan tilasta toiseen.



Kuva 24. Infoklusteri klusterianalyysin etupaneelissa.

LabVIEW-ohjelman pohjana käytetyssä MATLAB-ohjelmassa ei ollut kuvan haku -osiota, joten tämän osio rakennettiin alusta alkaen. Kuvassa 25 on esitetty Get Image -tila, joka hakee kuvan ja muokkaa sen käsiteltävään muotoon.



Kuva 25. Get Image -tila.

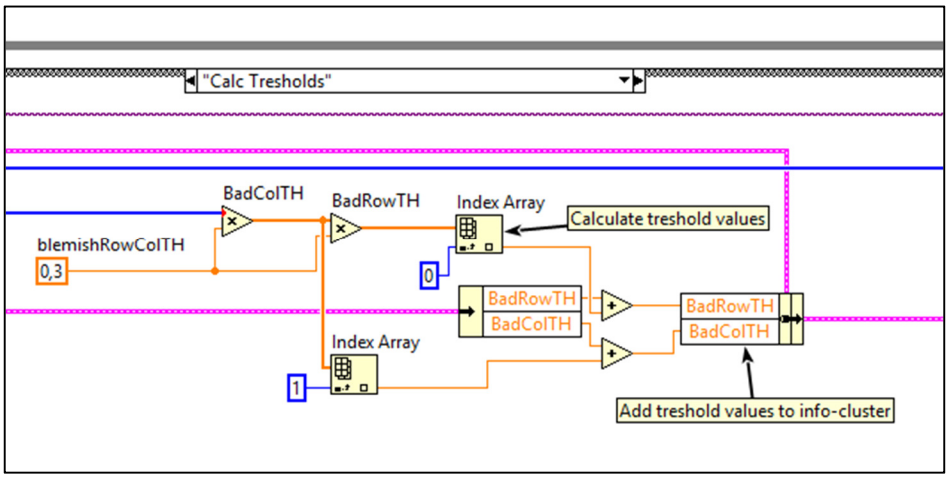
Tilassa on tavallisten LabVIEW-noodien lisäksi IMAQ-noodeja, jotka ovat osa Vision Acquisition Softwarea. IMAQ Create ja IMAQ ReadFile -noodeilla kuvalle luodaan tila muistiin ja luetaan kuva. Kuvasta haetaan parametrit ja kuva muunnetaan matriisi-muotoon, jotta sitä voidaan käsitellä. Valmis kuva siirretään Infoklusteriin, jossa data kul-jetetaan tilasta toiseen.

4.4.1 Calc Thresholds -tila

Tässä tilassa lasketaan blemishien eli virheellisten pikselien määrän raja-arvot riveissä ja sarakkeissa.

MATLAB-koodissa annettu raja (0,3) haetaan parametrit sisältävästä tiedostosta ja sijoi-tetaan muuttujiin badRowTH ja badColTH. Kuvan (blemishesGain) koko lasketaan `size`-funktiolla, joka palauttaa koon vektorimuodossa [pituus, korkeus] [36]. Muuttujaan sijoitetaan raja kerrottuna kuvan pituudella (badRowTH) tai korkeudella (badColTH).

LabVIEW-ohjelmassa (kuva 26) kuvalle lasketaan raja-arvot kaavoilla 0,3*kuvan pituus ja 0,3*kuvan korkeus. Arvot lisätään Unbundle-noodilla avattuihin BadRowTH- ja BadColTH-muuttujiin, jotka sijoitetaan Infoklusteriin. Raja-arvot ylittäneet rivit ja kolumnit analysoidaan ja muokataan myöhemmässä tilassa.



Kuva 26. Calc Thresholds -tila.

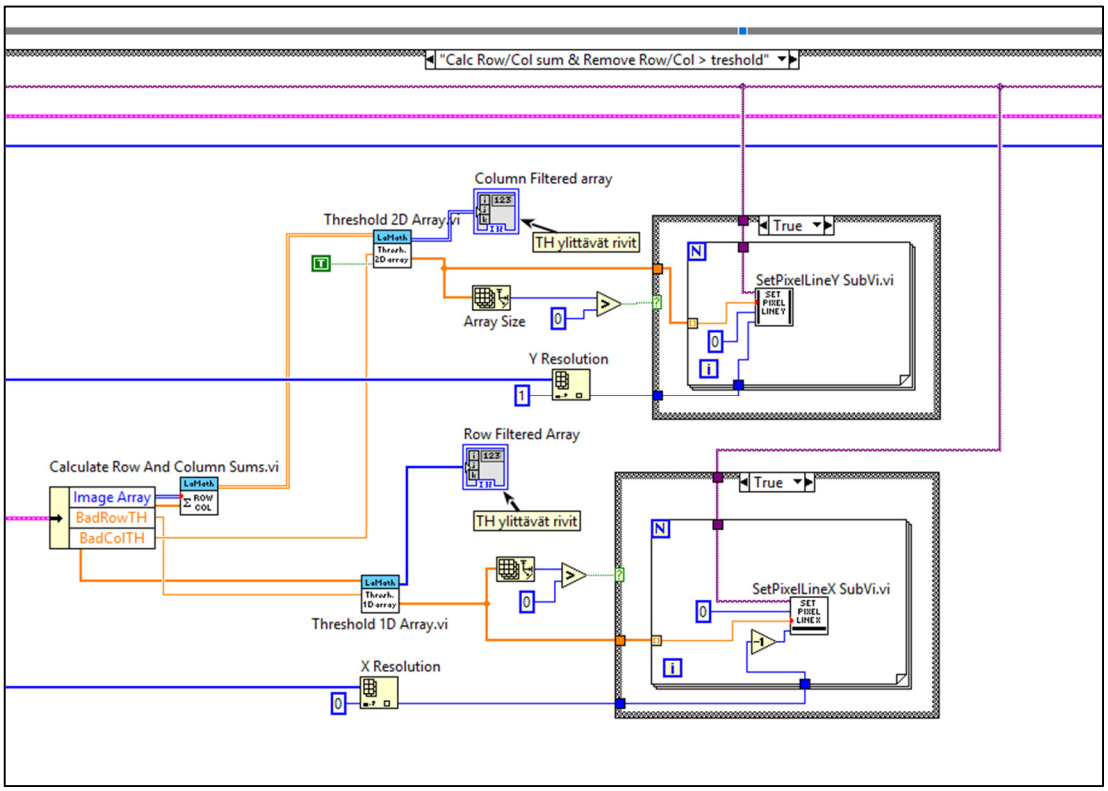
4.4.2 Calculate Row/Col sum & Remove Row/Col > Treshold -tila

Seuraavassa tilassa lasketaan blemishien rivi- ja sarakesummat ja poistetaan Calc Thresholds -tilassa määritetyn raja-arvon ylittävät rivit ja sarakkeet.

MATLAB-ohjelmassa lasketaan ensimmäisessä vaiheessa blemishien määrä kullakin rivillä ja sarakkeella käyttäen `sum`-funktiota, joka palauttaa elementtien (esimerkiksi pikseleiden arvojen) summan [37]. Kuva on tallennettu matriisin binäärimuodossa. Koska blemishit ovat 1:iä, Sum-funktion tuloksena on blemishien määrä. Tulokset tallennetaan BlemRowSum- ja BlemColSum-muuttujiin. Kun rivi- ja sarakesummat on laskettu, määritetään rivit ja sarakkeet, joissa blemishien summa on suurempi kuin kynnysarvo ja sijoitetaan ne blemishRows- ja blemishCols-muuttujiin. Määritetyt rivit ja sarakkeet sijoitetaan blemishesBobbleAnalysis-muuttuiin ja merkitään nollaksi.

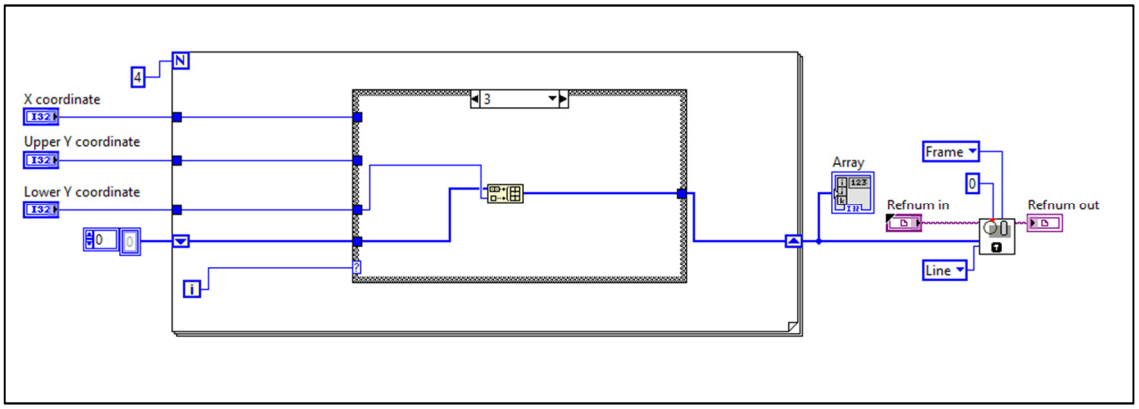
Kuvassa 27 on esitetty vastaava toteutettuna LabVIEW-koodina. Käyttäjä syöttää haluamansa kynnysarvot etupaneelilla sijaitsevaan BadRowTH- ja BadColTH-kontrolleihin. Kuva syötetään matriisimuodossa Calculate Row and Columns Sums.vi'hin, joka laskee ykkösten eli blemishien määrän riveillä ja sarakkeilla.

Sarakkeet syötetään Treshold 2D Array.vi'hin ja rivit Treshold 1D Array.vi'hin, jotka tulostavat rivit ja sarakkeet, joissa blemishien määrä ylittää kynnysarvon. Mikäli kynnysarvon ylittäviä rivejä/sarakkeita ei ole, ohjelma siirtyy seuraavaan tilaan. Jos kynnysarvon ylittäviä rivejä/sarakkeita on, huonojen rivien/sarakkeiden koordinaatit ja määrä syötetään for-silmukkaan, joka pyörii huonojen rivien/sarakkeiden määrän verran.



Kuva 27. Calculate Row/Col sum & Remove Row/Col > Treshold -tila.

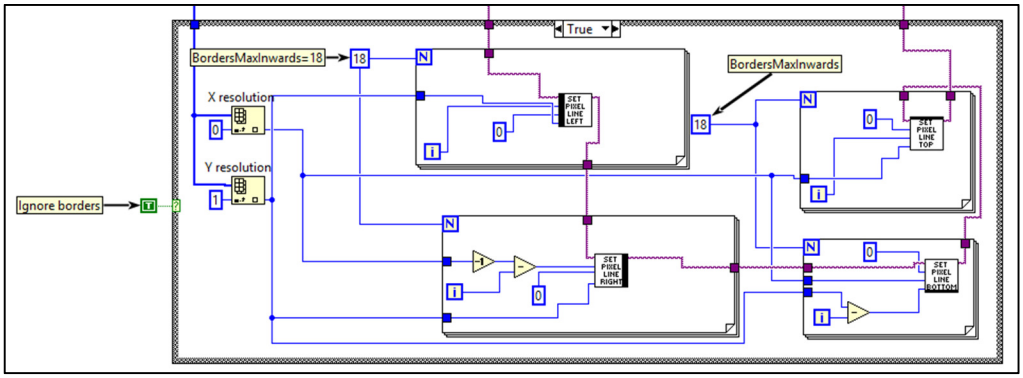
SetPixelLine(X/Y) -subVI muuttaa annettujen koordinaattien perusteella yksitellen rivin/sarakkeen pikselien arvoksi 0 kunnes kaikki koordinaatit on käyty läpi (kuva 28). Merkitäkseen rivin/sarakkeen pikselit nollaksi subVI vaatii alku- ja loppupisteen koordinaatit. Coordinate-kontrollit sisältävät subVI:hin syötetyt koordinaatit, jotka sijoitetaan oikeassa järjestyksessä matriisiin. IMAQ Draw -noodi piirtää matriisin avulla viivan arvolla 0 eli mustan viivan huonon rivin/sarakkeen päälle. Tämän jälkeen käsitelty kuva syötetään seuraavaan tilaan.



Kuva 28. SetPixelLine(X/Y) -subVI.

4.4.3 Remove blemishes from edges -tila

Käyttäjän halutessa ohjelma voi poistaa kuvan reunoista tietyn määrän pikseleitä eli niin sanotun reunahuntualueen; tämä tehdään Remove blemishes from edges -tilassa (kuva 29). Jos käyttäjä ei ole asettanut reunahuntualueen poistoa päälle (Ignore borders = False), tila ohitetaan.



Kuva 29. Remove blemishes from edges -tila.

Jos reunahuntualueen poisto on päällä (Ignore borders = True), poistetaan kuvan reunoista BordersMaxInwards verran pikseleitä. Kuvan X- ja Y-resoluutiot, sekä kuvan referenssi syötetään subVI:hin. Jotta mahdolliset ongelmat olisi helppo selvittää, jokaisella reunalla on oma subVI. Jokainen SetPixelLine-subVI on sijoitettu for-silmukkaan, jonka laskuriin on kytketty BordersMaxInwards tässä tapauksessa 18 kertaa, jolloin jokainen SetPixelLine-subVI poistaa pikselijonon 18 kertaa.

4.4.4 Standard Hough Transformation -tila

Intraoraalisensorien testikuvissa nähtiin monesti 45 asteen viivoja, joiden hakuun kuvasta käytettiin LabVIEW-ohjelman pohjana olleessa MATLAB-ohjelmassa Houghin muunnosta (osio 3.2.2). MATLAB:issa on tähän tarkoitukseen sisäänrakennettuja funktioita:

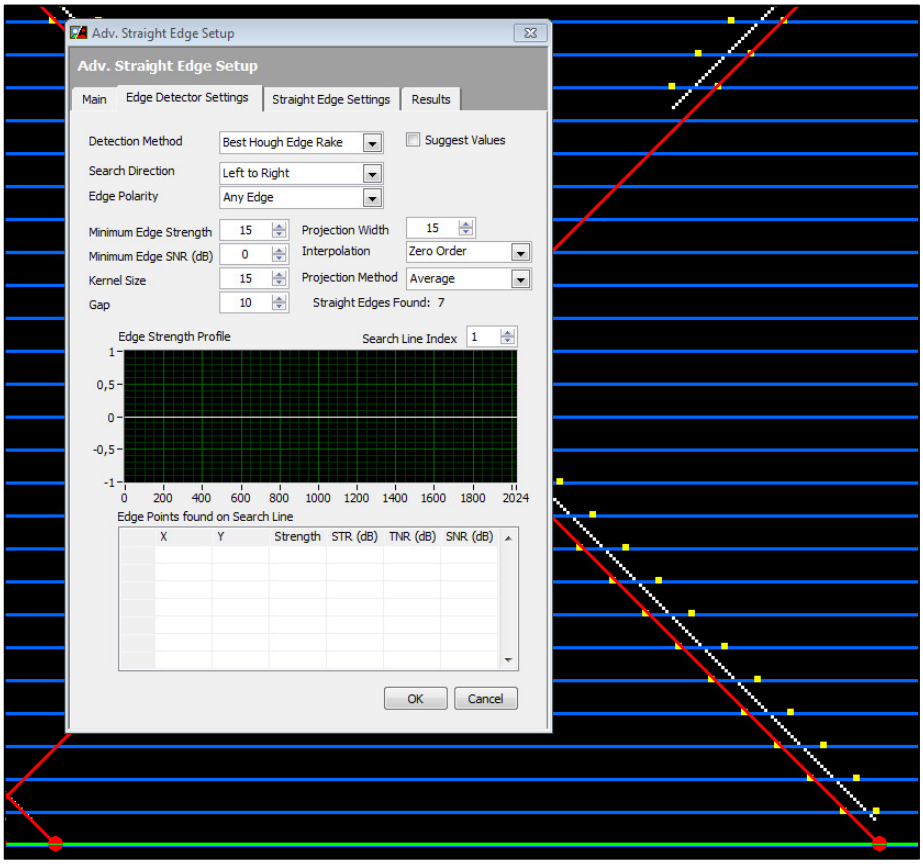
- `hough`-funktio `[H, theta, rho] = hough (BW)` suorittaa standardin Houghin muunnokseen binäärimuotoiselle kuvalle `BW`. Funktio käyttää Dudan ja Hartin kehittämää viivan parametrisoitua muotoa (kuva 10). Funktio palauttaa 1) `rho`:n, joka kuvaa viivan etäisyyttä origosta, 2) `theta`:n, joka kuvaa tämän vektorin ja x-akselin välistä kulmaa sekä 3) standardin Houghin muunnoksen `H`, joka on parametriavaruusmatriisi, jonka rivit vastaavat `rho`:ta ja sarakkeet `theta`:a. [38.]

- `houghpeaks`-funktio `peaks = houghpeaks(H,numpeaks)` hakee huippuja matriisista `H`. Huiput viittaavat viivan olemassaoloon kuvassa. Funktio palauttaa matriisin `peaks`, joka sisältää huippujen koordinaatit. [39.]

- `houghlines`-funktio `lines = houghlines(BW,theta,rho,peaks)` poimii viivasegmenttejä kuvasta `BW` [40].

MATLAB-ohjelma suorittaa kuvalle Houghin muunnoksen, jonka jälkeen palautetusta matriisista haetaan huippuja houghpeaks-funktiolla. Mikäli haun tulokset sisältävä matriisi on tyhjä, ohjelma käynnistää houghlines-funktion, joka hakee viivasegmenttejä. Mikäli ohjelma löytää 45 asteen viivoja, ohjelma muuttaa niiden pikseleiden arvoksi 0.

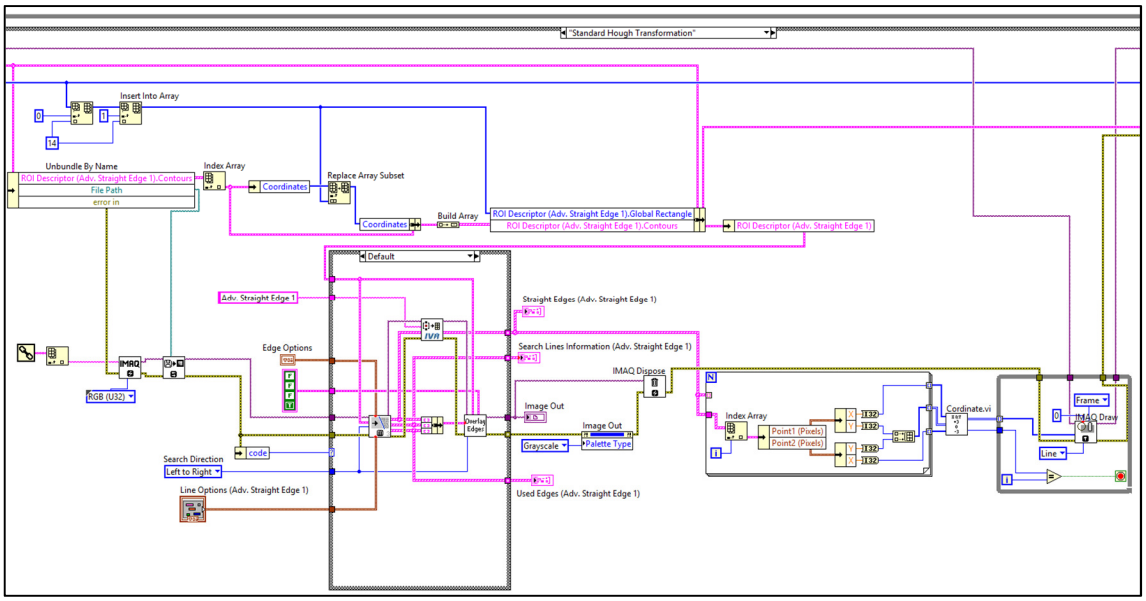
Toisin kuin MATLAB:issa, LabVIEW'ssä ei ole valmista funktiota Houghin muunnokselle. Planmeca hankki National Instrumentsin Vision Development Module -lisäosan kuvien käsittelyn tueksi. Lisäosaan kuuluvasta Vision Assistant-ohjelmasta löytyi Houghin muunnos Find Straight Edge -funktion alta Hough Rake -nimellä (kuva 30). Funktion tarkoitus on hakea reunoja, mutta se soveltuu myös viivojen hakuun.



Kuva 30. NI Vision Advanced Straight Edge -asetukset.

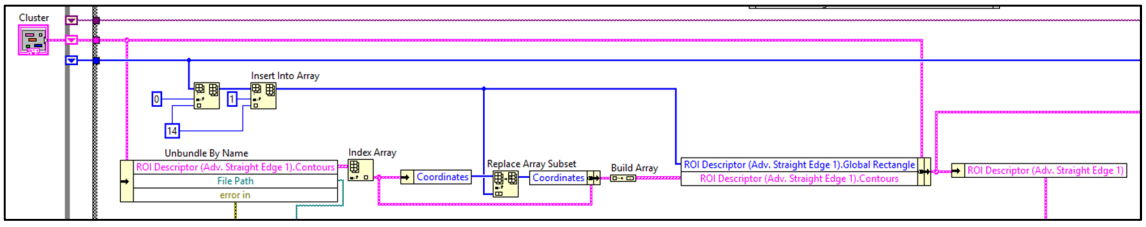
NI Vision Assitant määrittelee reunan merkittäväksi värin muutokseksi harmaasävyasteikolla vierekkäisten pikseleiden välillä. Sama määritelmä soveltuu myös viivojen hakuun. NI Visionin IMAQ Find Edge -työkalulla rajataan tutkittava alue. Hough-pohjainen reunanhaku hyödyntää Houghin muunnosta löytääkseen suoria viivoja. Ohjelma suorittaa muunnoksen ja palauttaa todennäköisimmät viivat. [26.]

Kun viivoja hakeva ohjelma oli valmis, ohjelma käännettiin LabVIEW VI:ksi. Ohjelma kopioitiin ja sovitettiin Klusterianalyysi VI:hin, Standard Hough Transform -tilaan (kuva 31).

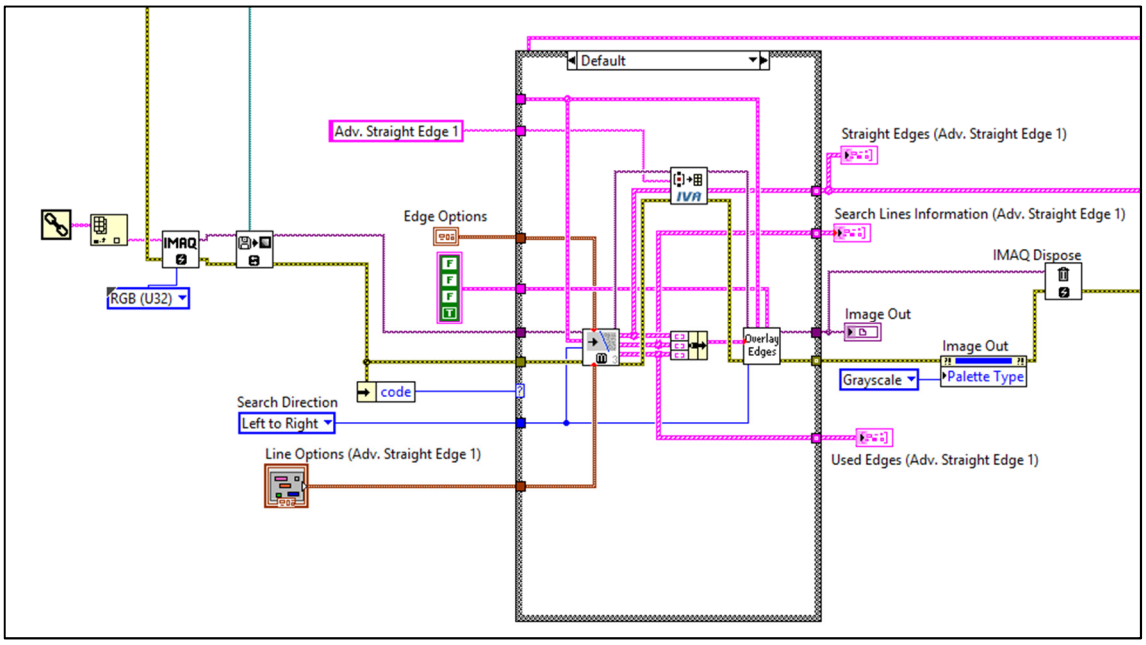


Kuva 31. Standard Hough Transform -tila kokonaisuudessaan.

Aluksi Infoklusteria puretaan Unbundle by Name -funktiolla, kunnes saadaan tutkittavan alueen (ROI, Region of Interest) koordinaatit muokattavaksi (kuva 32). Tutkittavaksi alueeksi merkitään koko kuva ilman aiemmin poistettua reunahuntualuetta.

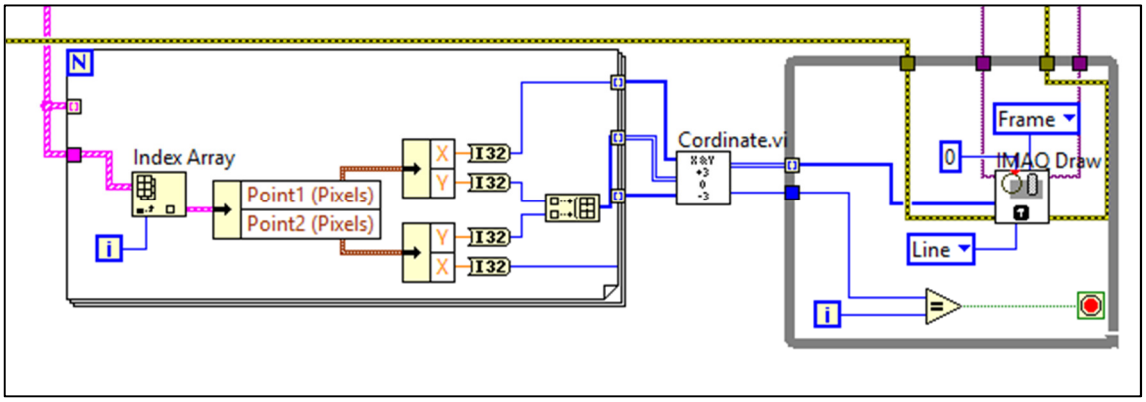


Kuva 32. Standard Hough Transformation -tila, alkuosa.



Kuva 33. Standard Hough Transformation -tila, keskiosa.

Valmiiksi muokatun kuva käyttö aiheutti virhetilan. Toimenpidettä varten ohjelma hakee kuvan uudestaan ja muuttaa sen RGB-muotoon. Tämän jälkeen uudelle kuvalle suoritetaan Houghin muunnos. Tulosten perustella IMAQ Find Straigh Edges 3 -subVI antaa Overlay Edges -noodille koordinaatit, joiden perusteella noodi piirtää punaisia viivoja RGB-kuvaan, joka näytetään etupaneelilla. Kuva poistetaan tämän jälkeen. IVA Store Particles Results -subVI siirtää punaisten viivojen alku- ja loppukoordinaatit eteenpäin käsiteltäväksi (kuva 33). Tulokset käsitellään for-silmukan sisällä (kuva 34).

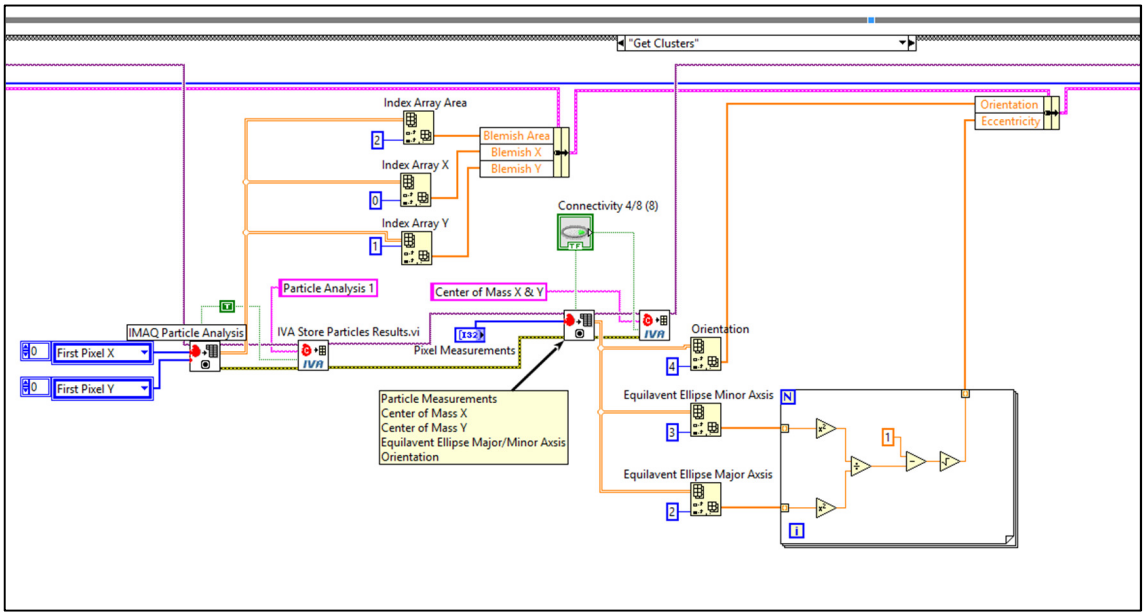


Kuva 34. Standard Hough Transformation -tila, loppuosa.

Punaiset viivat indikoivat poistettavan viivan olinpaikkaa. Niiden koordinaatit puretaan ylä-X- ja Y-koordinaateiksi ja ala-X- ja Y-koordinaateiksi. Punaisen viivan ja poistettavan viivan ero on pahimmillaan +/- 3 pikseliä. Coordinate-subVI piirtää koordinaattien perusteella 6 viivaa löydetyn viivan ympärille, jolloin virhealue käsitellään myös. IMAQ Draw -VI käsittelee alkuperäistä mustavalkoista kuvaa eikä Houghin muunnokseen ladattua kuvaa. VI piirtää mustan viivan Coordinate-subVI:n antamien koordinaattien välille peittäen viivablemishit (kuva 34).

4.4.5 Get Clusters -tila

Tässä tilassa blemisheistä haetaan ja tallennetaan rajaava alue, eksentrisyys ja suuntaus. MATLAB sisältää valmiin funktion `regionprops`, jolla voidaan laskea halutut parametrit. Tulokset tallennetaan vastaaviin muuttujiin. LabVIEW'ssä ei ole oletuksena noodia, jolla saataisiin suoraan blemishien halutut arvot eli koko, orientaatio ja eksentrisyys, minkä vuoksi käytettiin Vision Development Moduleen kuuluvia noodeja.



Kuva 35. Get Clusters -tila.

Koko lasketaan Vision Development Modulen IMAQ Particle Analysis.vi'lla ja IVA Store Particle Analysis Results.vi'lla, jotka hakevat blemishin ensimmäisen pikselin x- ja y-koordinaatit sekä blemishin koon (kuva 17). Orientaatio kuvaa kappaleen keskuksen läpi pienimmällä hitausmomentilla piirretyn viivan kulmaa. Orientaatio lasketaan vastapäi-

vään horisontaaliakselista ja sen arvo on 0–180 astetta. [41.] Orientaatio saadaan suoraan IMAQ Particle Analysis vi'sta [42]. Eksentrisyys on blemishin ympärille piirretyn kartioleikkauksen muotoa kuvaava parametri. Useimpien blemishien ympärille piirretty kartioleikkaus oli ympyrän muotoinen eli eksentrisyys oli 0. IMAQ Particle Analysis määrittää kartioleikkauksen laskuun tarvittavat parametrit pikkuakselin ja isoakselin pituudet. Eksentrisyys määritetään jokaiselle blemishille ja tallennetaan muuttujaan.

4.4.6 Get Largest Cluster -tila

Seuraavassa tilassa haetaan suurin klusteri, jonka ohjelma on löytänyt kuvasta. Klusteri koostuu useista blemisheistä jotka kostettavat toisiaan. Suuret klusterit ovat haitallisia sillä niiden alle voi jäädä piiloon tärkeitä yksityiskohtia.

MATLAB-ohjelma hakee suurimman klusterin `max`-funktiolla ja tallentaa sen parametrit muuttujaan.

LabVIEW'ssä klusterin koko ja X,Y -koordinaatit syötetään case-rakenteeseen. Ensimmäinen klusteri tallennetaan Shift Registeriin. Mikäli seuraava klusteri on suurempi kuin jo tallennettu, se tallennetaan aiemman klusterin yli. Ohjelma pyörii kunnes kaikki klusterit on käyty läpi. Suurimman klusterin parametrit tallennetaan lopuksi Infoklusterin Largest Cluster Area & X,Y -muuttujaan.

4.4.7 Save Image -tila

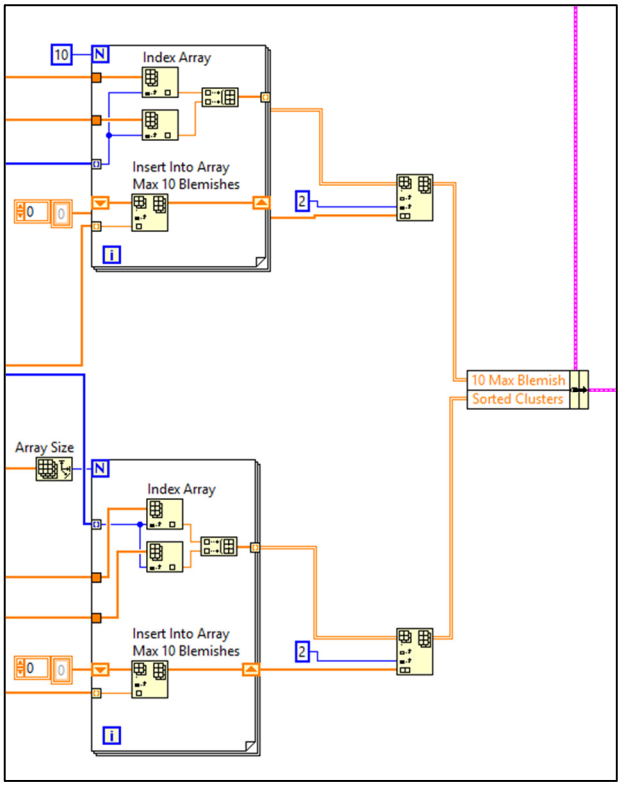
Tässä tilassa käsitelty kuva tallennetaan halutussa muodossa. MATLAB-ohjelma kääntää kuvan haluttuun muotoon ja tallentaa sen `imwrite`-funktiolla. LabVIEW'ssä tallennettava kuva käännetään ja tallennetaan esimerkiksi nimellä 'clusters.tiff'. LabVIEW'stä löytyy suoraan tallennusnoodi, johon syötetään tallennuspaikka ja kuvan referenssi. Tallennusnoodin tyyppi määrittää minkä tyyppinen tallennettu kuva on. Ohjelmassa käytettiin tiff-tyypin tallennusnoodia.

4.4.8 Sort Clusters, Save Biggest 10 -tila

Tässä tilassa klusterit järjestetään koon perusteella laskevaan järjestykseen ja tallennetaan 10 suurinta.

MATLAB sisältää valmiin `sort`-komennon, johon syötetään halututut elementit (aikaisemmassa tilassa käsitellyt blemishit) ja järjestys (laskeva). Klusterit tallennetaan ‘sortedBobbles’-muuttujaan. Näistä ensimmäiset 10 tallennetaan largestClusters-muuttujaan.

LabVIEW’ssä klusterit ja niiden koordinaatit järjestetään laskevaan järjestykseen ja syötetään kahteen for-silmukkaan. Ylempi for-silmukka tallentaa 10 suurinta klusteria alenevassa järjestyksessä 10 Max Blemish -muuttujaan. Alempi for-silmukka tallentaa klusterit ja niiden koordinaatit alenevassa järjestyksessä Sorted Clusters -muuttujaan (kuva 36).



Kuva 36. Sort Clusters, Save Biggest 10 -tila.

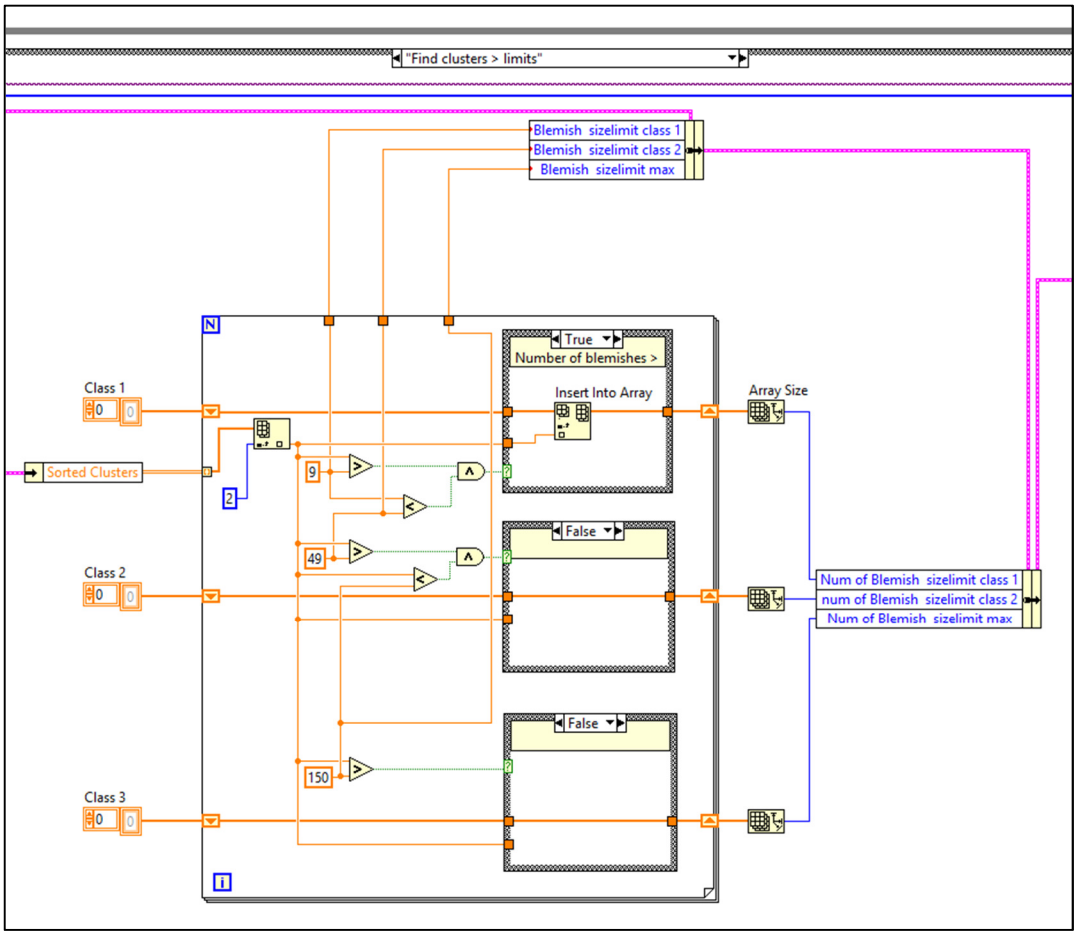
4.4.9 Find clusters > limits -tila

Ohjelma vertaa klustereita annettuihin kokorajoihin, joilla määritellään seuraavassa tilassa tarvitseeko intraoraalisensori mahdollisesti laaduntarkastuksen tai hylkäyksen.

MATLAB-ohjelmassa luodaan kolmiulotteinen matriisi [r,c,v], johon tulokset tallennetaan. Blemishejä verrataan config.cfg-tiedostosta haettuihin rajoihin 1,2 ja Max. Rajat ylittävät

klusterit tallennetaan matriisiin. Tallennetut klusterit jaetaan ryhmiin 1, 2 ja Max koon perusteella.

LabVIEW'ssä järjestetyt klusterit syötetään For-silmukkaan, joka pyörii kunnes kaikki klusterit on käsitelty. Silmukan sisällä klustereita verrataan raja-arvoihin. Ryhmän 1 raja on yli 9 pikseliä, ryhmän 2 raja on yli 49 pikseliä ja maksimiraja on yli 150 pikseliä. Klusterit tallentuvat kaikkiin niihin ryhmiin joiden rajan klusterin koko ylittää. Rajat ylittävien klustereiden määrä tallennetaan Num of Blemish sizelimit Class 1, 2 tai Max -muuttujaan Infoklusteriin (kuva 37).



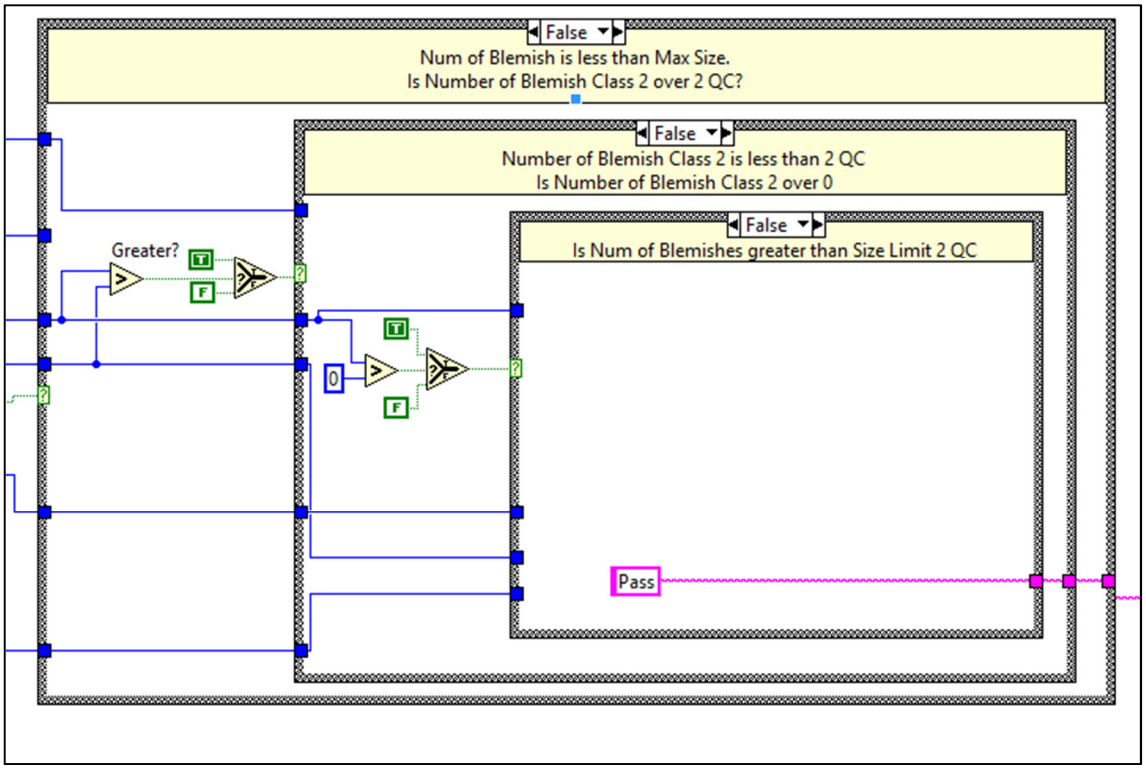
Kuva 37. Find clusters > limits -tila.

4.4.10 Compare clusters to limits -tila

Seuraavassa vaiheessa edellisessä tilassa määritettyjen kokorajat ylittäneiden klustereiden määriä verrataan rajoihin. Tulosten perusteella päätetään, tarvitseeko sensori laaduntarkastusta tai hylkäyksen.

MATLAB-ohjelma käy klusterit läpi yksitellen ja vertaa niitä config-tiedostosta haettuihin rajoihin. Mikäli raja-arvot ylittäviä klustereita on enemmän kuin sallittu, ohjelma tulostaa ylitettyä raja-arvoluokkaa vastaavan virheilmoituksen, esimerkiksi ”ReasonForQC = Too many 30/25 size 9-49 clusters”, mikä tarkoittaa että sensorille tulee suorittaa laaduntarkastus, koska siinä oli 30 klusteria kooltaan 9–49 pikseliä, kun sallittu määrä on 25. Sensori voidaan myös hylätä mikäli klustereita on liikaa tai siinä on lian suuri klusteri (yli 150 pikseliä).

LabVIEW-ohjelmassa vastaava vertailu tehdään Compare clusters to limits -tilassa (kuva 38) Tilassa on kaksi suurta case-rakennetta Class 1 ja Class 2, joiden sisällä on kaksi pienempää case-rakennetta. Raja-arvot ylittävät klusterit on aiemmassa tilassa jaettu ryhmiin Max, 1, 2, 1QC ja 2QC. Ryhmien koot syötetään suuriin case-rakenteisiin, mistä ne siirtyvät sisempiin case-rakenteisiin.



Kuva 38. Compare clusters to limits -tilan sisäkkäisiä case-rakenteita.

Molemmissa pää-case-rakenteissa verrataan Max-ryhmän kokoa Max-raja-arvoon. Jos klustereiden koko ei ylitä maksimiraja-arvoa, ohjelma siirtyy sisempään case-rakenteeseen, jossa verrataan 1- tai 2-ryhmän kokoa vastaavaan raja-arvoon. Maksimirajan ylittyessä sensori hylätään suoraan. Ohjelma tulostaa etupaneelille esimerkiksi ”Reason for

failing: 2 cluster(s) too large (size > 150)”. Viimeisessä case-rakenteessa 1 QC- tai 2 QC-raja-arvon ylitys johtaa korjaustoimenpiteisiin.

Lopuksi case-rakenteiden tulosteita verrataan. Jos Max/2/2QC-case-rakenteen tuloste on eri kuin ”Pass”, ohjelma tulostaa Max/2/2QC-case-rakenteen tulosteen. Tulosteen ollessa ”Pass” ohjelma tulostaa Max/1/1QC-case-rakenteen tulosteen. Näin molemmat case-rakenteet saadaan yhdistettyä Status-funktioon, joka tulostaa rajoihin vertaamisen tulokset etupaneeliin.

5 Yhteenveto

Tämän insinöörityön tavoitteena oli luoda uusia LabVIEW-ohjelmia Planmecan ProSensor HD -intraoraalisensorien testauslaitteelle, sekä kääntää olemassa olevia intraoraalisensorien testaussekvenssin MATLAB-ohjelmia LabVIEW-kielelle. Samalla kartoitettiin LabVIEW'n lisäosa Vision Development Modulen hyödyllisyyttä.

Testauslaitteen ohjelmointiprojektissa käytettiin Scrum-projektinhallintamenetelmää, jossa projekti jaetaan sprinteiksi kutsuttuihin jaksoihin. Sprintin alussa jaettiin tehtävät, jotka pyrittiin suorittamaan sprintin aikana. Jokaisen sprintin lopuksi pidettiin kokous, jossa käytiin läpi tulokset ja sovittiin tulevista tavoitteista. Scrum-menetelmään kuuluvat päivittäiset tapaamiset (daily scrum), joissa käydään läpi edistyminen ja potentiaaliset ongelmat. Scrum-menetelmä helpotti kommunikointia projektin jäsenten kesken ja projektin jakaminen pienempiin, aikataulutettuihin osiin tehosti työskentelyä. Aamutapaamisissa oli helppo kysyä mahdollisista ongelmista.

Projektin alussa toteutettiin kaksi yksinkertaista ohjelmaa, psComm-kommunikaattori ja RS-232-kommunikointi, joilla tietokone kommunikoi testauslaitteen kanssa. psComm-kommunikaattori ottaa vastaan käyttäjän komentoja ja muuttaa ne psComm-apuohjelman vaatimaan muotoon. Ohjelma myös muuttaa testauslaitteen psCommin kautta lähettämät vastaukset ymmärrettävään muotoon. Seuraavaksi luotiin ohjelma tietokoneen ja testauslaitteen väliseen RS-232-kommunikointiin käyttäen pohjana LabVIEW'n RS-232-kommunikointimallia. Käyttäjä asettaa kommunikointiparametrit ja -kanavan ja kirjoittaa etupaneelin Write-näyttöön haluamansa tekstin. Ohjelma muokkaa vastaukset ymmärrettävään muotoon; vastaus tulee Read-näyttöön.

Osana intraoraalisensorin testausta sensorilla otetaan testikuvia, joille suoritetaan joukko analyysejä. Analyysit oli aiemmin toteutettu MATLAB-ohjelmina, mutta testaussekvenssi haluttiin kääntää LabVIEW'lle. Alun perin insinöörityöprojektin puitteissa oli tavoitteena kääntää useampia analyysifunktioita LabVIEW'lle, mutta aika riitti vain Klusterianalyysin toteuttamiseen. Klusterianalyysissä tarkastellaan virheellisesti reagoineita pikseleitä eli blemishejä ja niiden muodostamia ryhmiä eli klustereita. Daily scrum -tapaamisessa päätettiin että Klusterianalyysi-ohjelmassa käytetään state machine -rakennetta. Vaikka käyttäjä ei kykene vaikuttamaan tilojen suoritusjärjestykseen, jako toisiaan seuraaviin tiloihin toiminnan perusteella selkeyttää ohjelmaa. Tiloihin jaettuun ohjelmaan

on myös helppo tehdä muutoksia joko muokkaamalla olemassa olevia tiloja tai lisäämällä uusia.

Valmis Klusterianalyysi-ohjelma koostuu 14 tilasta, joissa tutkitaan intraoraalisensorin testauksen tuloksena syntynyttä kuvaa, suoritetaan pieniä korjauksia ja verrataan tuloksia hylkyrajoihin. Lopuksi ohjelma tulostaa hylätäänkö sensori, vaatiiko sensori korjaustoimenpiteitä vai hyväksytäänkö sensori.

Sensorien testikuvissa oli ilmennyt 45 asteen viivoja, jotka ohjelman tuli löytää ja poistaa. Viivojen löytämiseksi kuvalle tuli suorittaa Houghin muunnos. MATLAB:ssa on sisäänrakennettuna funktio, joka suorittaa Houghin muunnoksen sekä muita Hough-funktioita. LabVIEW'ssä ei sen sijaan ollut oletuksena tarkoitukseen sopivia funktioita. Insinöörityöprojektin aikana Planmeca hankki LabVIEW'lle Vision Development Module -lisäosan, johon sisältyneestä Vision Assistant -ohjelmasta löytyi Houghin muunnos. Vision Assistantissa luodaan funktioista koostuva sekvenssi, joka on mahdollista muuntaa LabVIEW VI:ksi. Vision Assistantilla luotu algoritmi integroitiin Klusterianalyysin Standard Hough Transformation -tilaan ja muokattiin muuhun ohjelmaan sopivaksi. Standard Hough Transformation -tilassa kuvalle suoritetaan Houghin muunnos ja löydettyjen viivojen koordinaatit syötetään ohjelmaan, joka poistaa viivat koordinaattien perusteella. Viivojen haussa ilmeni pahimmillaan +/-3 pikselin virhe. Virhe korjattiin aluksi laajentamalla poistoaluetta, mutta ratkaisu ei ole optimaalinen. Virhealueeseen haetaan parempaa ratkaisua insinöörityöprojektin päättymisen jälkeen.

Vision Development Modulen mukana tuli Vision Assistant-ohjelman lisäksi myös IMAQ-noodeja, joita hyödynnettiin Klusterianalyysissä muun muassa kuvan hakuun ja käsittelyyn haluttuun muotoon. Vision Development Module -lisäosa nopeutti monin paikoin työntekoa. Erityisesti Houghin muunnoksen toteuttaminen olisi ollut huomattavasti vaikeampaa ilman Vision Development Modulea. Vaikka Klusterianalyysin viivojenhakuosion luominen LabVIEW'llä oli monimutkaista ja aikaa vievää, muiden osioiden kohdalla LabVIEW-ohjelmointi oli helpompaa ja lopullinen ohjelma oli usein yksinkertaisempi kuin MATLAB-versio.

Testikuvien analyysifunktioiden kääntäminen MATLAB:ista LabVIEW'lle sekä testauslaitteen muu ohjelmointi jatkuu insinöörityöprojektin päättymisen jälkeen. Työssä toteu-

tettuihin ohjelmiin tehdään todennäköisesti paranuksia ja muutoksia, jotta se ne soveltuvat paremmin osaksi testauslaitteen ohjelmistoa. Työn puitteissa toteutetut ohjelmat toimivat kuitenkin pääosin toivotulla tavalla.

Planmecan kannattaa tulevaisuudessakin panostaa LabVIEW-osaamiseen ja hyödyntää NI Visionia kuvankäsittelyä vaativissa sovelluksissa. LabVIEW-ohjelmoinnin perusteet on mahdollista oppia nopeasti, vaikka aikaisempi ohjelmointikokemus olisi vähäistä. LabVIEW'n G-kielen graafisen luonteen ansiosta LabVIEW-ohjelman toiminta on helppo hahmottaa, mikä helpottaa myös mahdollisten virheiden löytämistä.

Lähteet

1 Seeram, Euclid. 2011. Digital Radiography: An Introduction. Clifton Park, NY: Delmar Cengage Learning.

2 Wilhelm Conrad Röntgen – Biographical. 2014. Verkkodokumentti. <http://www.nobelprize.org/nobel_prizes/physics/laureates/1901/rontgen-bio.html>. Luettu 2.4.2017.

3 X-ray Production. 2016. Verkkodokumentti. <http://radiologykey.com/x-ray-production-2/>. Luettu 2.4.2017.

4 Whaites, Eric & Drage, Nicholas. 2013. Essentials of Dental Radiography and Radiology. Lontoo: Elsevier Health Sciences.

5 Seibert, Anthony J. 2005. Overview of Digital Detector Technology. Verkkodokumentti. <http://www.aapm.org/meetings/05am/pdf/18-2623-22086-53.pdf>. Luettu 5.2.2017.

6 Planmeca ProSensor HD -käyttöohje <http://materialbank.planmeca.com/#1485800140859_5>. Luettu 26.1.2017.

7 Suomalainen, Anni & Koskinen, Seppo K. 2013. Kartiokeilatietokonetomografia ja sen kliiniset sovellukset. Duodecim 129(10):1037–43.

8 Planmeca Group – asiantuntijavoimaa. Verkkodokumentti. Planmeca Oy. <http://www.planmeca.com/fi/Yritys/Planmeca-Group/>. Luettu 2.2.2017.

9 Better Care Through Innovation. Verkkodokumentti. Planmeca Group. <http://publications.planmeca.com/Brochures/Company/Planmeca_Group_en_low.pdf>. Luettu 15.1.2017.

10 Planmeca ProSensor. 2014. Verkkodokumentti. Planmeca USA, Inc. <http://www.planmeca.com/globalassets/usapdfs/intraoral/prosensor_brochure_51911.pdf>. Luettu 12.1.2017.

11 New Planmeca ProSensor® HD elevates the standard of intraoral dental imaging. Verkkodokumentti. Planmeca Oy. <http://www.planmeca.com/na/Press-info/News-room-main/new-planmeca-prosensor-hd-elevates-the-standard-of-intraoral-dental-imaging/>. Luettu 6.2.2017.

12 Planmeca Material Bank. Verkkosivu. Planmeca Oy. <http://materialbank.planmeca.com/#1491229626309_0>. Luettu 5.4.2017.

13 Learn About Scrum. Verkkodokumentti. Scrum Alliance. <https://www.scrumalliance.org/why-scrum>. Luettu 30.3.2017.

14 Larsen.W, Ronald. 2011. LabVIEW for Engineers. Boston: Prentice Hall.

15 LabVIEW Environment Basics. 2016. Verkkodokumentti. National Instruments. <http://www.ni.com/getting-started/labview-basics/environment> Luettu 25.3.2017.

16 Tutorial: SubVIs. 2015. Verkkodokumentti. National Instruments. <http://www.ni.com/white-paper/7593/en/>. Luettu 25.3.2017.

17 Execution Highlighting. 2012. Verkkodokumentti. National Instruments. <https://zone.ni.com/reference/en-XX/help/371361J-01/lvhowto/execution_highlighting/>. Luettu 25.1.2017.

18 Tutorial: State Machines. 2008. Verkkodokumentti. National Instruments. <http://www.ni.com/tutorial/7595/en/>. Luettu 24.1.2017.

19 Passing Data Between Loop Iterations in LabVIEW. 2014. Verkkodokumentti. National Instruments. <http://www.ni.com/getting-started/labview-basics/shift-registers>. Luettu 29.1.2017.

20 Case Structure. 2013. Verkkodokumentti. National Instruments. <http://zone.ni.com/reference/en-XX/help/371361K-01/glang/case_structure/>. Luettu 29.1.2017.

21 Using a State Machine (Event Driven) Architecture. 2006. Verkkodokumentti. National Instruments. <http://www.ni.com/white-paper/2926/en/>. Luettu 10.2.2017.

22 NI-VISA Overview. 2006. Verkkodokumentti. National Instruments. <http://www.ni.com/tutorial/3702/en/>. Luettu 14.2.2017.

23 What is the Difference Between NI Vision Acquisition Software, NI Vision Builder, and the NI Vision Development Module? 2007. Verkkodokumentti. National Instruments. <http://digital.ni.com/public.nsf/allkb/45A5682F377996BB862572B50072787F>. Luettu 31.8.2016.

24 What Is the Difference Between NI-IMAQ, NI-IMAQdx, and NI-IMAQ I/O? NI Vision Builder, and the NI Vision Development Module? 2011. Verkkodokumentti. National Instruments. <http://digital.ni.com/public.nsf/allkb/0564022DAFF513D2862579490057D42E>. Luettu 14.2.2011.

25 NI Vision Assistant Tutorial 2011. Verkkodokumentti. National Instruments. <http://www.ni.com/pdf/manuals/372228m.pdf>. Luettu 13.2.2017.

26 NI Vision 2011 Concepts Help.2011. Verkkodokumentti. National Instruments. <http://zone.ni.com/reference/en-XX/help/372916L-01/nivisioncon-cepts/edge_detection_concepts/>. Luettu 13.2.2017.

27 Antolovic, Danko. 2008. Review of the Hough Transform Method, With an Implementation of the Fast Hough Variant for Line Detection. Technical Report TR663. Indiana University.

28 Instrument Control in LabVIEW Tutorial. 2006. Verkkodokumentti. National Instruments. <http://www.ni.com/tutorial/3511/en/>. Luettu 15.3.2017.

29 Connect RS232/485 Serial Instruments to USB with the NI USB-232 and NI USB-485. 2006. Verkkodokumentti. National Instruments. <http://www.ni.com/white-pa2per/3146/en/232>. Luettu 22.2.2017.

30 Introduction to Serial Communications. Verkkodokumentti. TALtech. <http://www.taltech.com/datacollection/articles/serial_intro#synch>. Luettu 3.3.2017.

31 RS-232, RS-422, RS-485 Serial Communication General Concepts. 2016. Verkkodokumentti. National Instruments. <http://www.ni.com/white-paper/11390/en/>. Luettu 23.2.2017.

32 Overview of the Serial Port. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/matlab/matlab_external/overview-of-the-serial-port.html?s_tid=gn_loc_drop>. Luettu 4.3.2017.

33 Bies, Lammert. RS232 flow control and handshaking. Verkkodokumentti. <https://www.lammertbies.nl/comm/info/RS-232_flow_control.html>. Luettu 3.3.2017.

34 . Serial Settings:Flow Control. 2012. Verkkodokumentti. National Instruments. <https://zone.ni.com/reference/en-XX/help/371361J-01/lvvisaprop/serial_instr_3fff0025/>. Luettu 3.3.2017.

35 Termination Characters in NI-VISA. 2006. Verkkodokumentti. National Instruments. <http://www.ni.com/tutorial/4256/en/>. Luettu 6.3.2017Termination Characters in NI-VISA. 2006. Verkkodokumentti. National Instruments. <http://www.ni.com/tutorial/4256/en/>. Luettu 6.3.2017.

36 size. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/matlab/ref/size.html?searchHighlight=size>. Luettu 21.11.2016.

37 sum. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/matlab/ref/sum.html?searchHighlight=sum>. Luettu 23.11.2016.

38 hough. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/images/ref/hough.html?searchHighlight=hough>. Luettu 25.11.2016.

39 houghpeaks. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/images/ref/houghpeaks.htm>. Luettu 14.3.2017.

40 houghlines. Verkkodokumentti. MathWorks. <https://se.mathworks.com/help/images/ref/houghlines.html>. Luettu 14.3.2017.

41 Particle Measurements. 2016. Verkkodokumentti. National Instruments. <http://zone.ni.com/reference/en-XX/help/370281AC-01/nivisionconcepts/particle_measurements/>. Luettu 15.3.207.

42 IMAQ Particle Analysis Report VI. 2011. Verkkodokumentti <http://zone.ni.com/reference/en-XX/help/370281P-01/imaqvision/imaq_particle_analysis_report/>. Luettu 7.12.2.